НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
*Кафедра автоматизованих систем обробки інформації і управління*

УДК: 004.912

**«До захисту допущено»**
**В.о. завідувача кафедри**
_____ *О.А.Павлов*
(підпис) (ініціали, прізвище)
“___” _____2019 р.

# Дипломний проект
**на здобуття ступеня бакалавра**

з напряму підготовки *6.050101 «Комп’ютерні науки»*

на тему: « *Система класифікації текстів методами обробки природної мови та машинного навчання* »

**Виконав:** студент *4* курсу, групи *ІС-52*
*Юзьвак Дмитро Юрійович* _____
(прізвище, ім’я, по батькові) (підпис)

**Керівник** *доц., к.т.н. Сперкач М.О.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

**Консультант з графічної документації** *ст. викладач Халус О.А.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

**Рецензент** *доц., к.т.н., доц. Писаренко А.В.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент Юзьвак Д.Ю. _____
(підпис)

Київ – 2019 р.

# АНОТАЦІЯ

**Структура та обсяг роботи.** Пояснювальна записка дипломного проекту складається з шести розділів, містить 20 рисунків, 10 таблиць, 1 додаток та 11 джерел.

Дипломний проект присвячений вирішенню задачі класифікації текстів методами обробки природної мови та машинного навчання.

Метою створення системи є спрощення процесу класифікації текстів за рахунок створення моделей машинного навчання та оцінки їх ефективності.

У розділі загальних положень описано процес діяльності класифікації текстів, побудовано функціональну модель системи що проектується, визначено її відмінності від наявних аналогів. Визначено мету розробки та встановлено задачі, які необхідно вирішити.

У розділі інформаційного забезпечення надано детальний опис вхідних та вихідних даних, спроектовано базу даних для збереження результатів.

Розділ математичного забезпечення присвячений обґрунтуванню вибраних методів розв'язання задачі та опису алгоритмів на основі яких створюватимуться моделі машинного навчання.

Розділ програмного забезпечення описує засоби розробки програмного продукту та етапи проектування його архітектури. Описано специфікацію функцій та звіти, які генеруються в ході запуску програми.

У технологічному розділі визначено мету проведення випробувань програмного продукту та описано їх результати.

Результати проведеної роботи були опубліковані у міжнародному науковому журналі «Науковий огляд».

МАШИННЕ НАВЧАННЯ, ОБРОБКА ТЕКСТІВ, ОБРОБКА ПРИРОДНЬОЇ МОВИ, МОДЕЛЬ, КЛАСИФІКАЦІЯ ТЕКСТІВ.

## ABSTRACT

**The structure and the scope of the work.** Explanatory note of the diploma project consists of six sections, contains 20 drawings, 10 tables, 1 application and 11 sources.

The diploma project is devoted to solving the problem of classifying texts by methods of natural language processing and machine learning.

The section of the general provisions describes the process of classifying texts. The functional model of the system has been designed. System's differences from existing analogues were defined. The purpose of the development is determined and the tasks to be solved are set.

The purpose of the system is to simplify the process of classifying texts by creating models of machine learning and assessing their effectiveness.

The information section provides a detailed description of the input and output data, as well as a database for storing the results.

The section of mathematical support is devoted to the substantiation of the chosen methods of solving the problem and description of algorithms on the basis of which models of machine learning will be created.

The software section describes the software development tools and the stages of designing its architecture. The specification of functions and reports generated during program startup is described.

The technology section defines the purpose of testing the software product and describes their results.

The results of the work were published in the international scientific journal «Scientific Review».

MACHINE LEARNING, TEXT PROCESSING, NATURAL LANGUAGE PROCESSING, MODEL, CLASSIFICATION OF TEXTS.

# ЗМІСТ

# ВСТУП

З вибухом інформації підживлюється зростання світової мережі. Для людини все складніше стає аналізувати та обробляти дані, що надходять або навіть класифікувати дані за категоріями.

Більшість систем для видобування інформації (text mining) та її аналізу розраховані на те, що інформація вже представлена в зручній для аналізу формі (наприклад – базі даних). Однак, у більшості випадків, доступна в електронному вигляді інформація зберігається у вигляді неструктурованих документів, що відображають природню мову людини. Таким чином виникає проблема видобування інформації саме з неструктурованих систем, для чого і було розроблені методи обробки природньої мови [1].

Перед системами видобування інформації постає не тільки проблема структурування інформації, а й у тому, аби знайти спосіб представлення даних для подальшого аналізу та проведення будь-яких операцій із текстом.

Оскільки, більшість інформації (понад 80%) зберігається у вигляді текстів, вважається, що методи роботи з текстами мають високий комерційний потенціал. Знання можуть бути отримані («видобуті») з багатьох джерел інформації; та все ж, неструктуровані тексти залишаються найбільшим доступним джерелом знань [2].

Видобування інформації для структурування текстів стало захоплюючою науковою сферою, оскільки вона намагається виявити цінну інформацію з неструктурованих текстів. Неструктуровані тексти, які містять величезну кількість інформації, не можуть бути просто використані для подальшої обробки комп'ютерами. Таким чином, точні методи обробки, є важливими для того, щоб знайти цю інформацію.

Загалом, видобування тексту базується на наборі статистичних і комп'ютерних методів, що були спеціально розроблені для аналізу текстових

даних. Текст завжди був інформативним джерелом розуміння конкретної галузі чи окремих осіб. Тепер доступні нові джерела текстових даних, такі як текстові повідомлення, діяльність у соціальних мережах, блоги та веб-пошуки. Зростає доступність опублікованого тексту і також зростає інтерес до систем для вилучення інформації з тексту. Це в свою чергу призвело до заміни (або принаймні доповнення) людських зусиль автоматизованими системи обробки текстів.

Методи видобування тексту можна використовувати для різноманітних областей, починаючи від базових описів змісту тексту за допомогою слів і закінчуючи більш складними задачами, які стоять на перетині кількох областей комп'ютерних наук, таких як от класифікація текстів.

Зростання інформації і одночасне зростання доступної обчислювальної потужності комп’ютерів дозволяють використовувати сучасні методи для вирішення задачі класифікації.

Процес класифікації текстів займає досить багато часу, адже для вирішення цієї задачі необхідно не тільки прочитати текст, а й проаналізувати його зміст та обрати підходящу категорію із множини доступних. Коли ж мова йде про глобальні задачі та велику кількість даних, то часто виникають проблеми із недостачею людських ресурсів.

Класифікація неструктурованих текстів без використання автоматизованої системи полягає у тому, аби вручну визначити до якої категорії можна віднести текст. Кількість часу для вирішення цієї задачі прямо пропорційна кількості текстів, які необхідно класифікувати, та обернено пропорційна кількості експертів (людей, які вручну перечитують тексти та відносять їх до тих чи інших категорій). Саме тому виникає необхідність у створенні автоматизованих систем класифікації текстової інформації.

За останнє десятиліття було запропоновано безліч рішень цієї задачі. Однак, у більшості випадків, розв'язання задачі класифікації тестів пропонувалося для конкретного набору даних (наприклад, аби задовольнити бізнес-потреби компанії). Окрім цього було створено багато систем класифікації текстів, в основу яких лягли нейронні мережі.

Задача класифікації текстів, рішення якої буде доступним для широкого кола користувачів, все ще залишається відкритою. Саме з цією задачею працюватимемо в рамках виконання дипломного проекту.

**Практичне значення одержаних результатів**

Розроблено систему класифікації текстів методами природньої обробки мови та машинного навчання.

**Публікації**

Результати роботи були опубліковані у статті Юзьвак Д.Ю., Сперкач М.О. Розв'язання задачі класифікації текстів методами обробки природньої мови та машинного навчання/Юзьвак Д.Ю., Сперкач М.О.// Центр міжнародного наукового співробітництва «ТК Меганом» – Київ: 2019 (Подано до друку)

# 1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

## 1.1 Опис предметного середовища

З вибухом інформації підживлюється зростання світової мережі. Для людини все складніше стає аналізувати та обробляти дані, що надходять або навіть класифікувати дані за категоріями. При цьому зростання інформації і одночасне зростання доступної обчислювальної потужності комп'ютерів дозволяють використовувати сучасні методи для вирішення задачі класифікації.

Процес класифікації текстів займає досить багато часу, адже для вирішення цієї задачі необхідно не тільки прочитати текст, а й проаналізувати його зміст та обрати підходящу категорію із множини доступних. Коли ж мова йде про глобальні задачі та велику кількість даних, то часто виникають проблеми із недостачею людських ресурсів.

У дипломній роботі пропонується автоматизоване рішення задачі класифікації текстів.

Автоматизована класифікація текстів розглядається як життєво важливий метод для управління та обробки великої кількості цифрових документів, які широко поширені та постійно зростають. Загалом, класифікація текстів відіграє важливу роль у отриманні та підсумовуванні інформації, пошуку тексту та відповідей на запитання.

Пропонується розв'язати задачу класифікації текстів методами машинного навчання із застосуванням методів обробки природньої мови.

### 1.1.1 Опис процесу діяльності

Для того щоб вирішити задачу класифікації текстів необхідно пройти такі етапи:

- ознайомитися із доступними категоріями, на які можна розділити тексти;
- ознайомитися із переліком текстів та прочитати кожен текст;
- проаналізувати текст та віднести його до конкретної категорії.

Для того аби автоматизувати цей процес необхідно вирішити задачу машинного навчання. Задачі машинного навчання можна поділити на 2 типи: навчання з учителем (supervised learning) та навчання без учителя (unsupervised learning).

При навчанні з учителем машина знає результати роботу алгоритму ще до того як почне працювати з ним або вивчати його. При навчанні без учителя машина сама намагається зрозуміти суть алгоритму. Було вирішено вирішувати цю задачу саме методом з учителем, адже необхідно не просто поділити текстові документи на категорії, а й вказати конкретну категорію кожного документу. Для вирішення будь-якої задачі машинного навчання методом з учителем необхідно мати початковий набір даних (дата сет). В процесі автоматизації система працює з документами та аналізує їх зміст.

У кінці автоматизації ми отримаємо систему, що дозволятиме користувачеві самостійно створювати моделі машинного навчання на основі власних дата сетів, навчати їх та аналізувати їх ефективність.

Розглянемо схему структурну процесу діяльності, що представлена у розділі графічного матеріалу.

Опишемо процес діяльності класифікації текстів.

Увійшовши в систему, користувач має можливість створити модель машинного навчання, або переглянути моделі, які були створенні раніше.

Якщо користувач обирає перегляд раніше створених моделей, то система автоматично з’єднується із базою даних та завантажує дані про моделі машинного навчання, які були створені раніше. На цьому етапі користувач має можливості переглянути ефективності створених моделей

(точність із якою та чи інша модель класифікує конкретний текст), а також запустити модель та протестувати її шляхом завантаження конкретних текстів.

Якщо користувач обирає створення нової моделі, то йому пропонується завантажити набір раніше класифікованих текстів (дата сет) у систему. Одразу ж перевіряється коректність завантаженого файлу. Зауважимо, що дата сет повинен бути представлений у форматі *.csv, *.tsv або *.xls.

Якщо формат не вірний, користувач отримає повідомлення про помилку, інакше дата сет буде завантажено у систему та відображено користувачеві.

На наступному етапі користувач має вказати, яка колонки із дата сету відповідає текстам, а яка категоріям. Це робиться із метою уникнення помилок у випадку коли дата сет складається із більшої кількості колонок.

Наступним етапом має бути обробка природньої мови текстів із дата сету. Заважимо, що цей етап займає багато часу, тому було прийняте рішення зберігати результати роботи цього етапу в окремому файлі. Таким чином, якщо користувач використовуватиме дата сет, що був завантажений раніше, система автоматично визначить це та завантажить в систему файл із підготовленими до класифікації текстами.

Цей етап прихований від користувача і він виконується у системі автоматично. Таку таблицю називають корпусом слів. Саме її будемо зберігати в окремому файлі з метою економії часу при подальшій роботі з дата сетом.

На наступному етапі обираються алгоритми, які використовуватимуться для створення моделей машинного навчання. Таким алгоритмами є: алгоритм логістичної регресії, наївний Байєсів алгоритм, алгоритм опорних векторів або ж алгоритм дерева ухвалення рішень.

Після того, як моделі машинного навчання були створені, необхідно їх зберегти, для подальшої роботи з ними. Після цього етапу виконується етап аналізу ефективності моделей машинного навчання, який детальніше розглянемо у наступних розділах. Вся інформація про створені моделі записується до бази даних.

На наступному етапі користувачеві пропонується класифікувати конкретний текст на основі створених моделей машинного навчання. Для цього необхідно завантажити текст у систему, класифікувати його за допомогою обраних алгоритмів та вивести результат класифікації тексту на екран. Після цього робота системи завершується.

### 1.1.2 Опис функціональної моделі

Опишемо функціональну модель системи за допомогою схеми структурної варіантів використання, що представлена на рисунку 1.1.

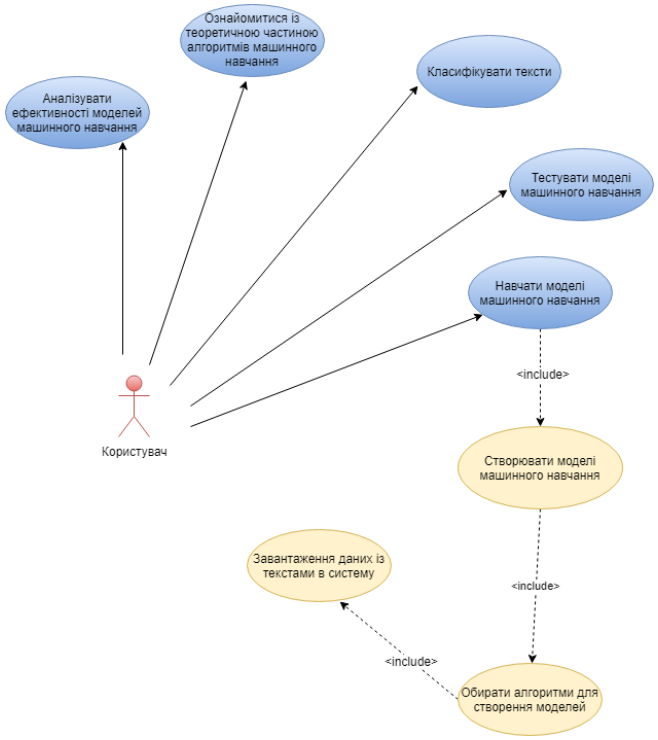



Рисунок 1.1 – Схема структурна варіантів використання

Розглянемо, які функції може виконувати користувач в рамках автоматизованої системи класифікації текстів:

- завантажувати дата сет в систему;
- ознайомитися із теоретичною частиною алгоритмів машинного навчання;
- обирати алгоритми для створення моделей машинного навчання;
- створювати моделі машинного навчання;
- навчати та тестувати моделі машинного навчання;
- аналізувати ефективності моделей машинного навчання;
- класифікувати тексти.

Для того аби забезпечити користувача вищенаведеними функціями, система має виконувати такі функції:

- завантаження дата сету в систему;
- аналізування дата сету;
- підготовка текстів до навчання (методами обробки природньої мови);
- створення моделей машинного навчання;
- оцінка моделей машинного навчання;
- класифікація текстів із використанням створених моделей.

## 1.2 Огляд наявних аналогів

Проаналізуємо наявні аналоги даної системи. Детальний опис аналогів представлено у таблиці 1.1.

Таблиця 1.1 - Огляд наявних аналогів

| Назва | Алгоритми | Техно-логії | Суть задачі | Дата сет | Посилання |
|---|---|---|---|---|---|
| Toxic Comment Challenge | Logistic Regression Convolutional neural network model BiDirectional LSTM (Recurrent neural networks) | Python 2.7, Tensorflow 1.4, SciPy, Numpy. | The purpose of this project is to build models that best classifies the comments into 6 categories | Kaggle | https://github.com/arunarn2/ToxicCommentChallenge |
| Text Classification | Naive Bayes Linear Classifier Support Vector Machine (SVM) Deep Neural Networks | Python 2.7, Tensorflow 1.4, SciPy, Numpy. | Divide text into 2 labels | Amazon reviews | https://www.analyticsvidhya.com/blog/2018/04/a-comprehensive-guide-to-understand-and-implement-text-classification-in-python/ |
| Multi-Class Text Classification with Scikit-Learn | Logistic Regression (Multinomial) Naive Bayes Linear Support Vector Machine Random Forest | Python 2.7, SciPy, Numpy. | Divide text into 12 categories | Consumer complaints | https://datascienceplus.com/multi-class-text-classification-with-scikit-learn/ |
| Spam Classification | K-Nearest Neighbor, Decision tree, Random Forest, Logistic regression | NLP, Python 2.7, SciPy, Numpy. | Spam or not | Built in dataset | https://github.com/eduonix/nlptextclassification |
| Text Classification | Logistic Regression Support Vector Machine | NLP, R | Classify BBC articla into 5 categories | BBC dataset | https://www.stat.berkeley.edu/~aldous/Research/Ugrad/Nura_Kawa_report.pdf |

Розглянемо особливості кожного програмного продукту, що був приведений у таблиці 1.1.

Toxic Comment Challenge базується на нейронній мережі, що дозволяє класифікувати коментарі за наведеними категоріями. На вхід програмного продукту подається короткий текст, а на виході отримуємо категорію, до якої

нейронна мережа класифікувала коментар.

Text Classification базується на моделях машинного навчання, що дозволяють класифікувати текст на 2 категорії.

Multi-Class Text Classification with Scikit-Learn базується на моделях машинного навчання, що дозволяють визначити чи є текст потенційним спамом чи ні.

Text Classification дозволяє класифікувати статті BBC за 5-ма категоріями. Програмний продукт написаний на мові програмування R

Однією із головних відмінностей програмного продукту, що розробляється є те, що користувач самостійно може обрати необхідні дані, на яких треба навчити модель, обрати алгоритми та переглянути результати навчання. Всі програмні продукти які я знайшов дозволяють користувачеві лише отримати кінцевий результат.

Програмний продукт, що розробляється дозволяє пройти процес навчання моделі самостійно, отримати оцінку кожної із моделей машинного навчання та обрати алгоритм, що найкраще підходить для вирішення конкретної задачі. Таким чином можна зробити висновок що розробка даного програмного продукту є актуальною, оскільки жоден із аналогів не пропонує користувачеві кілька варіантів вирішення задачі, а одразу приводить до кінцевого результату.

## 1.3 Постановка задачі

### 1.3.1 Призначення розробки

Призначенням розробки є класифікація текстів методами обробки природньої мови та машинного навчання.

### 1.3.2 Мета та задачі розробки

Метою розробки є спрощення процесу класифікації текстів за рахунок створення моделей машинного навчання із застосуванням методів обробки природної мови.

Завдяки цьому користувачі можуть створювати власні моделі на обраних масивах інформації (дата сетах із раніше класифікованими текстами), тестувати їх при різних умовах, а також оцінювати ефективність кожної із моделей.

Для досягнення мети необхідно вирішити такі задачі:

- підготовка дата сетів для тестування моделей машинного навчання;
- первинна обробка текстів із застосуванням методів обробки природньої мови (NLP – natural language processing);
- створення моделей машинного навчання на основі обраних алгоритмів;
- навчання моделей на підготовлених дата сетах;
- оцінка ефективності моделей;
- класифікація текстів;
- візуалізація проведених експериментальних досліджень.

### Висновок до розділу

У ході написання даного розділу дипломного проекту було описано предметне середовище, яке розглядається. Встановлено, що людині все складніше обробляти велику кількість інформації та вирішувати задачі пов’язані із класифікацією, а зростання її кількості та одночасне зростання доступної обчислювальної потужності комп’ютерів дозволяють автоматизувати розв’язання задачі класифікації.

Окрім цього було описано процес діяльності, який автоматизується, визначено, які етапи потрібно пройти, аби розв’язати задачу, наведено

коротку характеристику методів, якими пропонується вирішувати поставлену задачу. Було побудовано діаграму діяльності та наведено її детальний опис.

Також, описано функціональну модель системи та визначено функції користувача, а також проведено аналіз наявних аналогів та встановлено, які відмінності матиме продукт, що розробляється.

Сформовано призначення розробки, встановлено мету та визначено задачі, які необхідно вирішити для досягнення мети.

# 2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

## 2.1 Вхідні дані

Для вирішенні задачі класифікації текстів необхідно підготувати матеріал, для навчання моделей машинного навчання. У нашому випадку таким матеріалом виступають підготовлені набори даних, що у термінах машинного навчання називаються дата сетами.

Приклад такого дата сету для подальшої класифікації текстів представлено у таблиці 2.1.

Таблиця 2.1 – Приклад дата сету

| Категорія | Текст |
|---|---|
| Sport | Claxton hunting first major medal<br>British hurdler Sarah Claxton is confident she can win her first major medal at next month's European Indoor Championships in Madrid.<br>The 25-year-old has already smashed the British record over 60m hurdles twice this season, setting a new mark of 7.96 seconds to win the AAAs title. "I am quite confident," said Claxton. "But I take each race as it comes. "As long as I keep up my training but not do too much I think there is a chance of a medal." Claxton has won the national 60m hurdles title for the past three years but has struggled to translate her domestic success to the international stage. Now, the Scotland-born athlete owns the equal fifth-fastest time in the world this year. And at last week's Birmingham Grand Prix, Claxton left European medal favourite Russian Irina Shevchenko trailing in sixth spot. |
| Politics | Tony Blair has apologised to two families who suffered one of the UK's biggest miscarriages of justice.<br>The prime minister was commenting on the wrongful jailing of 11 people for IRA bomb attacks on pubs in Guildford and Woolwich in 1974. Mr Blair said: "I am very sorry that they were subject to such an ordeal and injustice." He made the apology to members of the Conlon and Maguire families in his private room at Westminster. In a statement recorded for television, Mr Blair said the families deserved |

Продовження таблиці 2.1

| | |
|---|---|
| | "to be completely and publicly exonerated".<br>The families had hoped the apology would be made during Prime Minister's Questions in the House of Commons. However, one of the so-called Guildford Four, Gerry Conlon - who was wrongly convicted of planting the bombs - said the families were delighted with the apology. He said Mr Blair had spoken with "such sincerity", adding: "He went beyond what we thought he would, he took time to listen to everyone. "You could see he was moved by what people were saying. "Tony Blair has healed rifts, he is helping to heal wounds. It's a day I never thought would come." The move followed a huge campaign in Ireland for a public apology after eleven people were wrongly convicted of making and planting the IRA bombs which killed seven people. Mr Blair's official spokesman said no-one present at the meeting would "ever forget the strength of feeling of relief that the prime minister's statement brought to them". |
| Business | Battered dollar hits another low.<br>The dollar has fallen to a new record low against the euro after data fuelled fresh concerns about the US economy.<br>The greenback hit $1.3516 in thin New York trade, before rallying to $1.3509. The dollar has weakened sharply since September when it traded about $1.20, amid continuing worries over the levels of the US trade and budget deficits. Meanwhile, France's finance minister has said the world faced "economic catastrophe" unless the US worked with Europe and Asia on currency controls. Herve Gaymard said he would seek action on the issue at the next meeting of G7 countries in February. Ministers from European and Asian governments have recently called on the US to strengthen the dollar, saying the excessively high value of the euro was starting to hurt their export-driven economies. "It's absolutely essential that at the meeting of the G7 our American friends understand that we need coordinated management at the world level," said Mr Gaymard.<br>Thursday's new low for the dollar came after data was released showing year-on-year sales of new homes in the US had fallen 12% in November - with some analysts saying this could indicate problems ahead for consumer activity.<br>Commerce Department data also showed consumer spending - which drives two thirds of the US economy - grew just 0.2% last month. |

Окрім цього, є ще інші вхідні дані, але вони не задаються користувачем, а визначаються програмно. У наступних версіях програмного продукту планується надати користувачам можливість створювати моделі машинного навчання в режимі адміністратора та змінювати параметри моделі. Такими вхідними даними є:

- коефіцієнт розмежування дата сету;
- номер максимального значення параметрів класифікації.

Розглянемо кожен із вищенаведених параметрів детальніше.

**Коефіцієнт розмежування дата сету**

Однією із поставлених задач є не тільки навчання моделей, а й їх тестування. Для того щоб протестувати модель, необхідно знати, наскільки точно вона класифікує тексти. Для цього нам потрібна тестова вибірка. Для того, щоб її отримати достатньо розділити початковий дата сет на 2 вибірки: навчальну і тестову. Відсоткове співвідношення цього розділення і є коефіцієнтом розмежування дата сету.

Коефіцієнт розмежування дата сету лежить в інтервалі [0; 1]

Наприклад, якщо в початковому дата сеті міститься 10 000 текстів, то коефіцієнт розмежування тексту, рівний 0.4 означає, що 40% (4 000 текстів) складатимуть тестову вибірку, а 60% (6 000 текстів) складатимуть навчальну вибірку.

Експериментально встановлено, що коефіцієнт розмежування дата сету, який даватиме найкращі результати лежить в межах [0.15; 0.25]. У початковій версії програмного продукту ми приймемо це число сталим і рівним 0.2.

Варто зазначити, що занадто великий коефіцієнт розмежування дата сету приведе до неточностей при навчанні моделей, а занадто малий не дасть конкретної оцінки ефективності моделі машинного навчання.

**Номер максимального значення параметрів класифікації**

Для того, щоб створити модель машинного навчання необхідно представити тексти у форматі, що буде зрозумілим машині. Цей етап називається обробкою природньої мови. В результаті його робити ми отримуємо таблицю, що дає інформацію про наявність слів у тому чи іншому тексті.

Розглянемо приклад. Нехай маємо 2 тексти: “He was quite sure about this decision” і “My mother made a quite good decision”.

Результатом створення так званого корпусу слів для цих двох текстів буде така таблиця:

Таблиця 2.2 – Приклад корпусу слів

| Слово | Decision | Quite | Sure | He | Mother | Make | Good |
|---|---|---|---|---|---|---|---|
| Текст1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| Текст2 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |

Такі слова як ‘the’, ‘was’, ‘about’, ‘this’, ‘a’ не несуть в собі ніякого силового відтінку, отже їх можна не представляти моделі машинного часу із ціллю економії часу.

Із таблиці видно, що у тестах присутні слова на перетині із колонками яких стоїть значення 1.

Кількість колонок, в цій таблиці відповідає номеру максимального значення параметру класифікації. У даному випадку це 7. У реальних дата сетах із великою кількістю даних це число може доходити до кількох мільйонів, тому з метою економії часу на навчання моделі та економії дискового простору прийнято зменшувати це число на 20-35%, відкидаючи слова, що найрідше зустрічаються у моделі.

## 2.2 Вихідні дані

Під час запуску програми користувач може виконувати певні функції, що були описані у попередньому розділі. Внаслідок роботи цих функцій створюються вихідні дані. Вихідними даними є:

- створена модель машинного навчання, що зберігається у вигляді файлу із машинним кодом (приклад такого файлу наведено на рисунку 2.1);
- файл із інформацією про ефективність створеної моделі.

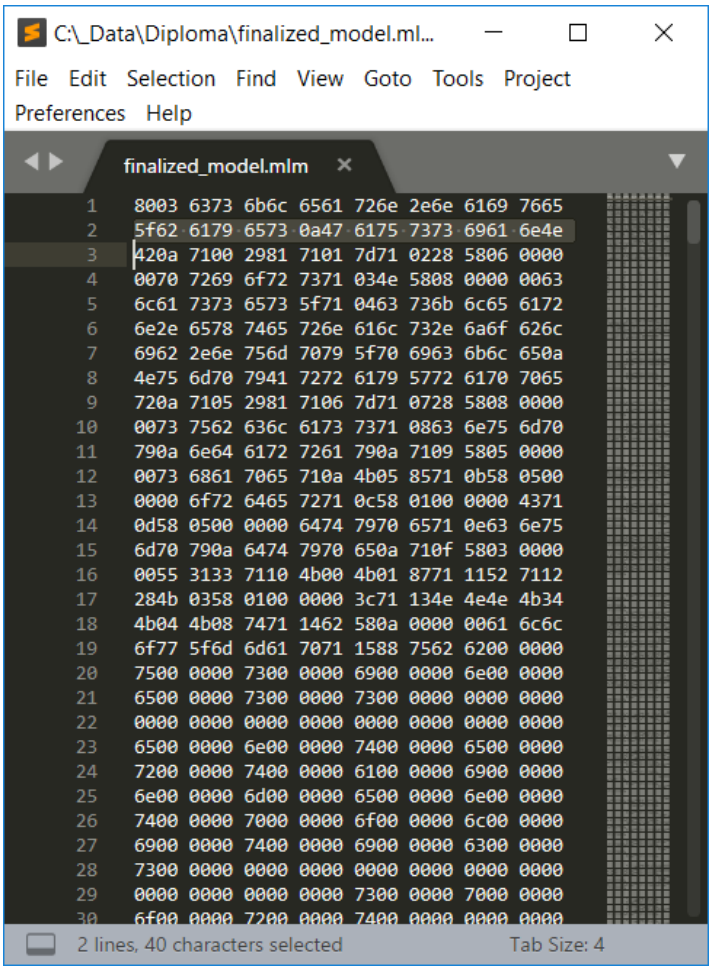



Рисунок 2.1 – Приклад машинного коду збереженої моделі машинного навчання

## 2.3 Опис структури бази даних

Розглянемо схему бази даних, що представлена на рисунку 2.1.


Рисунок 2.2 – Схема бази даних

Із рисунку видно, що база даних складається із двох таблиць: інформації про модель машинного навчання та інформації про дата сет, на основі якого була створена та чи інша модель. Зв'язок між цими таблицями: один до багатьох. Це означає те, що на основі одного дата сету може бути створено декілька моделей машинного навчання.

Детальний опис таблиці із інформацією про модель наведено у таблиці 2.3.

Таблиця 2.3 – Детальний опис таблиці БД із загальною інформацією

| Поле | Тип | Опис |
|---|---|---|
| id | integer <pk> | Унікальний номер запису в таблиці. |
| dataset_id | integer | Номер запису із таблиці з дата сетами. |
| name | text | Назва моделі відповідає поєднанню назви дата сету та назві алгоритму. |
| algorithm | text | Назва алгоритму на основі якого була створена модель машинного навчання. |
| creation date | date | Дата створення моделі зберігається в цілях відстеження завантаження системи. |
| dataset_name | text | Назва дата сету. |
| accuracy | double precision | Точність моделі – значення з інтервалу [0; 1], яке вказує наскільки ефективно модель класифікує тексти. |
| file_with_model | text | Посилання на файл моделі, що був збережений одразу після її створення. |
| file_with_info | text | Посилання на файл із інформацією про модель машинного навчання. |

Детальний опис таблиці із інформацією про дата сет наведено у таблиці 2.4.

Таблиця 2.4 – Детальний опис таблиці БД із інформацією про ефективність моделі

| Поле | Тип | Опис |
|---|---|---|
| id | integer <pk> | Унікальний номер запису в таблиці. |
| file_with_dataset | text | Посилання на файл із дата сетом. |
| records | integer | Кількість записів у дата сеті. |

Продовження таблиці 2.4

| Поле | Тип | Опис |
|---|---|---|
| train_records | integer | Кількість записів у дата сеті для тренування моделей машинного навчання. |
| test_records | integer | Кількість записів у дата сеті для тестування моделей машинного навчання. |
| number_categories | integer | Кількість категорій у дата сеті. |
| categories | text | Перелік категорій. |

У таблиці 2.3 із загальною інформацією міститимуться такі поля: унікальний номер запису, номер запису із таблиці з дата сетами, назва моделі, назва алгоритму, дата створення, назва дата сету, точність моделі, посилання на файл моделі, посилання на файл з інформацією про модель.

У таблиці 2.4 із інформацією про ефективність моделі машинного навчання містяться такі поля: унікальний номер запису, посилання на файл із дата сетом, кількість записів у дата сеті, кількість записів для тренування моделей, кількість записів для тестування моделей, кількість категорій у дата сеті, перелік категорій.

**Висновок до розділу**

У ході написання даного розділу були визначені вхідні та вихідні дані системи, що розробляється.

Надано детальний опис вхідних даних, що не задаються користувачем, а визначаються програмно, залежно від обраного алгоритму для створення моделей.

Окрім цього, було наведено приклад створення корпусу слів, що є важливим етапом обробки природньої мови.

Встановлені параметри створення моделей машинного навчання.

Також, проаналізовано доцільність використання бази даних для збереження необхідної інформації про моделі машинного навчання. Спроектовано модель бази даних та її структуру. Наведено детальний опис атрибутів таблиць бази даних, визначено їх тип.

# 3 МАТЕМАТИЧНЕ ЗАБЕЗЕЧЕННЯ

## 3.1 Змістовна постановка задачі

Уведемо позначення, які будуть зустрічатися при розв'язанні задачі класифікації тестів методами обробки природньої мови та машинного навчання.

Текст – загалом зв'язна і повністю послідовна кінцева множина слів.

Слово – найменша самостійна і вільно відтворювана в мовленні відокремлено оформлена значима одиниця мови, набір символів [3].

Нехай маємо набір текстів, що були класифікованими за певними категоріями. Такі тексти для зручності наводяться у вигляді таблиці 3.1.

Таблиця 3.1 – Представлення текстів

| Тексти | Категорії |
|---|---|
| Текст 1 | Категорія 1 |
| … | … |
| Текст k-1 | Категорія 1 |
| Текст k | Категорія 2 |
| … | … |
| Текст *l* | Категорія 2 |
| … | … |
| Текст t | Категорія m |
| … | … |
| Текст n | Категорія m |

Задача полягає у тому, аби однозначно визначити, до якої категорії відноситься текст, що не належить набору текстів, тобто не був класифікованим раніше.

Розглянемо приклад такої задачі. Зауважимо, що даний приклад є незрівнянно малим у порівнянні із реальними задачами, які будемо розглядати у рамках даної дипломної роботи.

Нехай маємо 10 текстів, що відповідають відгукам про ресторан. Тексти класифіковані за такою ознакою позитивний відгук, негативний відгук. Дата сет наведено у таблиці 3.2.

Таблиця 3.2 – Приклад задачі для класифікації текстів

| Текст | Категорія |
|---|---|
| On the up side, their cafe serves really good food. | Позитивний відгук |
| The only good thing was our waiter, he was very helpful and kept the bloody Mary's coming. | Позитивний відгук |
| Good prices. | Позитивний відгук |
| Although I very much liked the look and sound of this place, the actual experience was a bit disappointing. | Негативний відгук |
| Worst service to boot, but that is the least of their worries. | Негативний відгук |
| For a self-proclaimed coffee cafe, I was wildly disappointed. | Негативний відгук |
| Overall, I was very disappointed with the quality of food. | Негативний відгук |
| At first glance it is a lovely bakery cafe - nice ambiance, clean, friendly staff. | Позитивний відгук |
| Anyway, I do not think i will go back there. | Негативний відгук |
| Those burgers were amazing. | Позитивний відгук |

Задача полягає у тому, аби класифікувати будь-які відгуки, що не належать даному дата сету за однією із двох категорій: позитивний відгук, негативний відгук.

Розглянемо математичну постановку задачі та методи якими цю задачу можна розв'язати. Саме розв'язання задачі наведемо у розділі 3.5.

### 3.2 Математична постановка задачі

Нехай маємо множину текстів *D*, що складається із довільних текстів:

$$D = \{d_1, d_2, ...d_n\}, \quad (3.1)$$

де $d_i$ – конкретний текст, $i = \overline{1,n}$, $n$ – кількість текстів.

Нехай маємо множину категорій $C$, на які можна розділити дані тексти:

$$C = \{c_1, c_2, ...c_m\}, \tag{3.2}$$

де $c_i, i = \overline{1,m}$ конкретний текст, $m$ – кількість категорій.

Множина текстів $D$, кожен елемент якої класифікований по заданим категоріям із множини $C$ називається початковий дата сетом, або просто дата сетом, тобто набором даних. Дата сет позначатимемо як $Ds$.

$$Ds = \begin{pmatrix} d_1 & c_1 \\ \vdots & \vdots \\ d_{k-1} & c_1 \\ d_k & c_2 \\ \vdots & \vdots \\ d_l & c_2 \\ \vdots & \vdots \\ d_t & c_m \\ \vdots & \vdots \\ d_n & c_m \end{pmatrix} \tag{3.3}$$

Дата сет побудований за допомогою невідомої цільової функції $F$:

$$F : C \times D \to \{0,1\} \tag{3.4}$$

Задача класифікації текстів методами машинного навчання полягає у тому, аби побудувати класифікатор $F`$, максимально наближений до $F$.

Класифікатором $F`$ зветься визначена на множині текстів $D$, функція яка однозначно зіставляє кожен текст із множини $D$ конкретній категорії із множини $C$.

Для вирішення цієї задачі ми використовуватимемо методи машинного навчання у поєднанні із обробкою природних мов. Планується розглядати такі алгоритми машинного навчання: Наївний Баєсів класифікатор (Naive

Bayes Classifier), Метод логістичної регресії (Logistic Regression), Дерево ухвалення рішень (Decision Tree) та Метод опорних векторів (Support Vector Machine).

### 3.3. Опис методів розв'язання

#### 3.3.1 Опис алгоритму логістичної регресії

Логістична регресія є методом підбору лінії регресії $y = f(x)$, у випадку коли у складається із даних, що можна представити у двійковому вигляді. Якщо відповідь, що необхідно отримати є двійковою (дихотомічною) змінною, а x є числовим значенням, то логістична регресія відповідає логістичний кривій функціонального відношення між $x$ та $y$ [5].

Перевагою класифікатора є простота реалізації.

Недоліками є невисока якість класифікації, значні обчислювальні витрати та факт того, що додавання текстів у тренувальну вибірку може значно вплинути на результат обчислення вагових коефіцієнтів, а значить і змінити характеристики моделі.

#### 3.3.2 Опис алгоритму наївного Байєса

Це простий імовірнісний класифікатор, заснований на застосуванні теореми Байєса з сильними (наївними) припущеннями про незалежність [6].

Перевага наївного Байєсівого класифікатора полягає в тому, що він вимагає лише невеликої кількості навчальних даних для оцінки необхідних параметрів класифікації.

Класифікатор Naive Bayes є, мабуть, найпростішим і найбільш широко використовуваним класифікатором. Він моделює розподіл документів у кожному класі з використанням імовірнісної моделі, припускаючи, що розподіл різних термінів незалежний один від одного [6, 8].

Перевагами класифікатора є простота реалізації та низькі обчислювальні витрати. Недоліками є невисока якість класифікації.

### 3.3.3 Опис алгоритму опорних векторів

Метод опорних векторів – це алгоритм, який визначає у якому місці необхідно розмежувати класи векторів, які належать певним групам. Він може бути застосований до будь-яких векторів, які кодують будь які дані. Найпростіше розглянути цей алгоритм у Декартовому просторі. Вирішуючи задачу класифікації текстів та перетворюючи тексти у вектори ми стикаємося із багатовимірними просторами, з якими легко справляється комп'ютер, але які важко зрозуміти людині [6].

Перевагами класифікатора є його ефективність на великих об’ємах даних та простота реалізації. До недоліків слід віднести високі обчислювальні витрати.

### 3.3.4 Опис алгоритму дерева ухвалення рішень

Дерево рішень є ієрархічним деревом навчальних екземплярів, в якому умова значення атрибуту використовується для поділу даних ієрархічно. Іншими словами, дерево рішень рекурсивно розділяє набір навчальних даних на менші підрозділи на основі набору тестів, визначених у кожному вузлі або гілці. Кожен вузол дерева є тестом, і кожна гілка, що спускається з вузла, відповідає одному значенню цього атрибута. Екземпляр класифікується, починаючи з кореневого вузла, перевіряючи атрибут цим вузлом і рухаючись вниз по гілці дерева відповідає значенню атрибута в даному екземплярі. І цей процес рекурсивно повторюється. У випадку текстових даних, умови на вершині дерева вершин зазвичай визначаються термінами в текстових документах. Наприклад, вузол може бути поділений на своїх дітей, спираючись на наявність або відсутність конкретного терміну в документі.

Крім цього, варто зазначити, що існує багато методів, які як окремо так і в поєднанні один з одним можуть значно покращувати ефективність роботи алгоритму.

Переваги класифікатора є: ефективний на великих об’ємах даних, не потребує параметрів для навчання, інтуїтивно зрозумілий алгоритм, може бути представлений у вигляді скінченного набору правил. Недоліками є те, що алгоритм легко “перевчити” на тестових даних. Оскільки дерева рішень використовують метод «поділяй і володарюй», вони, як правило, працюють добре, якщо існують декілька високо релевантних атрибутів, але менше, якщо присутні багато складних взаємодій. Тобто із зростанням кількості класів ефективність алгоритму погіршуватиметься [7].

### 3.4 Обґрунтування методів розв’язання

#### 3.4.1 Обґрунтування алгоритму логістичної регресії

У роботі [5] наведено детальний опис роботи цього алгоритму. Розглянемо його основні положення. Проста лінійна регресія може бути представленою формулою 3.5:

$$y = \frac{e^x}{1+e^x} = \frac{1}{1+e^{-x}}\,. \qquad (3.5)$$

Графік цієї залежності представлено на рисунку 3.1.

Рисунок 3.1 – Логістична регресія

Щоб забезпечити гнучкість, логістична формула може бути представлена у наступному вигляді:

$$y = \frac{e^{\alpha+\beta x}}{1+e^{\alpha+\beta x}} = \frac{1}{1+e^{-(\alpha+\beta x)}}, \qquad (3.6)$$

де $\alpha$ та $\beta$ визначають логістичний перетин і нахил.

Логістична регресія відноситься до типу експоненційних класифікаторів. Як і наївний Байєсів класифікатор, логістична регресія працює через знаходження коефіцієнтів ваг, які множаться на вхідні ознаки та лінійно з'єднує їх.

Найбільш вагомою різницею між логістичною регресією і наївним Байєсовим класифікатором є те, що регресія є дискримінативною (тобто такою, що складається із окремих частин), а наївний Байєсев класифікатор генеративним (тобто використовує правило Байєса). Загалом,

дискримінативна модель моделює межі між класами (категоріями), а генеративна моделює фактичний розподіл кожного класу. Фактично обидві моделі шукають умовну ймовірність [5].

Лінійна регресія дає можливість безпосередньо обчислити умовну ймовірність P(B|A), а точніше передбачає якою вона буде, основуючись на початковому дата сеті. Логістична регресія передбачає цю ймовірність обираючи деякий набір вхідних даних, з'єднуючи їх лінійно (множачи кожен вхідний сигнал на якусь вагу та додаючи доданки) та застосовуючи функцію (3.6) до отриманого результату. Однак, ми не можемо просто обчислити ймовірність безпосередньо:

$$P(y = C \mid x) = \sum_{i=1}^{N} w_i f_i = wf \qquad (3.7)$$

У такому випадку ми отримаємо значення із всієї множини дійсних чисел. Ніщо із рівняння 3.7 не змушує результат лежати на проміжку від 0 до 1. Для того, аби вирішити цю проблему необхідно додати незалежний знаменник, щоб нормалізувати отриманий результат та звузити його до інтервалу [0; 1].

Таким чином значення ймовірності обчислюватимемо за формулою (3.8):

$$P(c \mid x) = \frac{\sum_{i=1}^{N} w_i f_i(c, x)}{\sum c' \in e^{\sum_{i=1}^{N} w_i f_i(c, x)}} \qquad (3.8)$$

Задача класифікації тепер зводиться до пошуку вагових коефіцієнтів. Їх пошук відбувається на основі початкового (тренувального) дата сету. Тобто, машина підбирає такі коефіцієнти, щоб класи тестових текстів підходили самим текстам із найбільшою ймовірністю. Наприклад, якщо при ваговому коефіцієнті 5 ймовірність віднесення тексту про освіту до категорії «Освіта» дорівнює 0.54, а при ваговому коефіцієнті 4 ймовірність віднесення тексту

про освіту до категорії «Освіта» дорівнює 0.78, то очевидно, що для цього прикладу необхідно обрати коефіцієнт 4. Проводячи подібні розрахунки на мільйонах даних, в результаті ми отримуємо класифікатор, який з високою точністю відноситиме тексти до потрібних категорій.

Таким чином ми обираємо такі параметри вагових коефіцієнтів, які максимізують логарифмічну функцію ймовірностей для вибірки із тренувальними значеннями.

Для кожного спостереження $x^{(j)}$ в тренувальній вибірці оптимальними вагами є:

$$\widehat{w} = \arg\max_{w} \log P(y^{(i)} \mid x^{(i)}). \qquad (3.9)$$

Для повної вибірки даних оптимальні ваги визначаються за формулою:

$$\hat{w} = \arg\max_{w} \sum_{j} \log P(y^{(i)} \mid x^{(j)}). \qquad (3.10)$$

Після знаходження вагових коефіцієнтів класифікатор побудований за алгоритмом лінійної регресії можна використовувати для класифікації текстів.

### 3.4.2 Обґрунтування алгоритму наївного Байєса

У роботі [8] наведено детальний опис роботи цього алгоритму. Розглянемо його основні положення. Основна суть наївного Байєсового класифікатора полягає у тому, що для будь-якого документа $d$, що може бути віднесений до будь-якої категорії із $C$ класифікатор повертає клас $\widehat{c}$, що має максимальну апостеріорну ймовірність.

Апостеріорна ймовірність – умовна ймовірність, яка вираховується після врахування певних умов, що можуть впливати на її значення [9].

$$\widehat{c} = \arg\max_{c \in C} P(c \mid d) \qquad (3.11)$$

Ця ідея класифікатора Байєса відома ще з часів роботи Байеєа (1763) і була вперше використана для класифікації тестів Мостеллером і Уоллесом у

1964 році. [4]. Суть цього класифікатора полягає у тому, аби перетворити формулу (3.10) за допомогою формули Байєса (3.11) у рівняння, що допоможе позбутися умовної ймовірності, замінивши її значеннями інших ймовірностей:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}. \qquad (3.12)$$

Тоді формула 3.11 приймає вигляд:

$$\hat{c} = \arg\max_{c \in C} P(c|d) = \frac{P(d|c)P(c)}{P(d)}. \qquad (3.13)$$

Можна значно спростити формулу (3.12), шляхом відкидання знаменника *P(d)*. Це можна зробити, адже ми рахуватимемо значення цієї ймовірності для кожної категорії із множини *C*. А це значення із знімною категорії залишатиметься незмінним. Тобто, нам необхідно отримати значення ймовірності для тексту *d*, а оскільки сам текст не змінюватиметься, то і ймовірність залишатиметься такою самою. Отже, отримали спрощену формулу:

$$\hat{c} = \arg\max_{c \in C} P(c|d) = \arg\max_{c \in C} P(d|c)P(c). \qquad (3.14)$$

Таким чином, ми шукаємо найбільш ймовірний клас $\hat{c}$, який отримуємо в результаті аналізу конкретного тексту. Значення цієї ймовірності складається із добутку двох величин: $P(c|d)$– ймовірності (likelihood) та $P(c)$ так званої попередньої ймовірності (prior probability).

Зауважимо, щоб без втрати узагальнення ми можемо представити документ $d$ у вигляді набору ознак (наприклад, ознак присутності слів у тексті) $f_1, f_2, ..., f_t$.

Тоді формула 3.13 приймає вигляд:

$$\hat{c} = \arg\max_{c \in C} P(f_1, f_2, ..., f_n|d)P(c). \qquad (3.15)$$

На жаль, формулу 3.14 все ще дуже складно обчислити безпосередньо, без деяких спрощуючих припущень, оцінюючи ймовірність кожної ознаки. Цей процес потребує величезного набору вхідних даних та займає багато часу. Тому ми зробимо 2 речі, що допоможуть спростити процес пошуку ймовірності. На першому кроці створимо торбу слів (the bag of words).

Торба слів (the bag of words) – набір всіх слів у тексті. Тобто для речення «Tom loves his mom» торба слів виглядатиме: Tom, loves, his, mom. Таким чином ми враховуватимемо лише наявність слова у тексті і упускатимемо його позицію чи кількість однакових слів.

На другому кроці зробимо так зване наївне Байєсівське припущення, від якого власне і походить назва методу. Це припущення говорить: припустимо, що ймовірності впливу кожного слова на результат є незалежними, тоді

$$P(f_1, f_2, ..., f_t \mid c)P(c) = P(f_1 \mid c)P(c) \cdot P(f_2 \mid c)P(c) \cdot ... \cdot P(f_t \mid c)P(c). \qquad (3.16)$$

Таким чином ми спростили формулу 3.15:

$$\hat{c} = \arg\max_{c \in C} P(f_1, f_2, ..., f_t \mid c)P(c) = \arg\max_{c \in C} P(c) \prod_{f \in F} P(f \mid c). \qquad (3.17)$$

Таким чином, використовуватимемо формулу 3.17 для класифікації текстів класифікатором наївного Байєса.

Перед тим, як класифікувати тексти, класифікатор необхідно навчити на раніше підготовлених класифікованих текстах. Опишемо цей процес з математичної точки зору.

Для того щоб обчислити ймовірність $P(c)$ та $P(f_i \mid c)$ спершу розглянемо максимальну прогнозовану ймовірність. Для цього використовуватимемо частоти в даних. Для кожного класу $c \in C$ обчислимо, який відсоток документів знаходиться у класі $c$. Нехай $Nc$ – кількість документів, що належать класу $c$, а $N$ - загальна кількість документів. Тоді

$$P(c) = \frac{Nc}{N}. \qquad (3.18)$$

Щоб знайти ймовірність $P(f_i \mid c)$ розглядатимемо ознаки присутності слів у тексті. Тобто $P(w_i \mid c)$ – ймовірність того, що випадково вибране слово із множини слів для всіх документів із категорії с виявиться словом $w_i$. Тобто, для кожної категорії необхідно побудувати множину слів $V$ і для кожного слова із $V$ обчислити частоту його появи у цій множині. Таким чином ми побудували наївний Байєсів класифікатор.

### 3.4.3 Обґрунтування алгоритму опорних векторів

У роботі [4] наведено детальний опис роботи цього алгоритму. Розглянемо його основні положення.

Розглянемо метод опорних векторів для вирішення задачі класифікації двох текстів. При збільшенні кількості категорій суть методу не змінюватиметься, змінюватиметься лише кількість просторів.

Нехай маємо деякі вхідні дані: $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$, де $x_i \in R^n$, $y_i \in \{+1, -1\}$.

Функція рішення для методу опорних векторів виглядає як

$$g(x) = \operatorname{sgn}(f(x)), \qquad (3.19)$$

$$f(x) = \sum_{i=1}^{n} y_i a_i K(x_i, x) + b, \qquad (3.20)$$

де $K$ – функція ядра, $b \in R$ – порогове значення і $a_i, i = \overline{1,n}$ – вагові коефіцієнти.

Зауважимо, що вагові коефіцієнти задовольняють таким умовам

$$\forall i : 0 \le a_i \le C \qquad (3.21)$$

та

$$\sum_{i=1}^{N} y_i a_i = 0, \qquad (3.22)$$

де $C$ – вартість неправильної класифікації. Вектори $x_i$ називаються опорними векторами.

Для лінійного методу опорних векторів функція ядра К визначена як

$$K(x_i, x) = x_i, x. \qquad (3.23)$$

Тоді рівняння 3.20 приймає вигляд

$$f(x) = wx + b, \qquad (3.24)$$

де $w = \sum_{i=1}^{n} y_i a_i x_i$.

Для того, аби навчити модель побудовано на цьому алгоритмі класифікувати тексти необхідно знайти вагові коефіцієнти, тобто вирішити задачу оптимізації, що представлена виразом 3.25:

$$\sum_{i=1}^{n} a_i - \frac{1}{2} \sum_{i,j=1}^{n} a_i a_j y_i y_j K(x_i, x_j) \to \max \qquad (3.25)$$

при обмеженнях (3.21) та (3.22).

Розв'язок цієї задачі для класифікації двох текстів дасть оптимальну гіперплощину, яка розділяє 2 класи текстів (представлених векторами). Таке розділення представлене на рисунку 3.2.

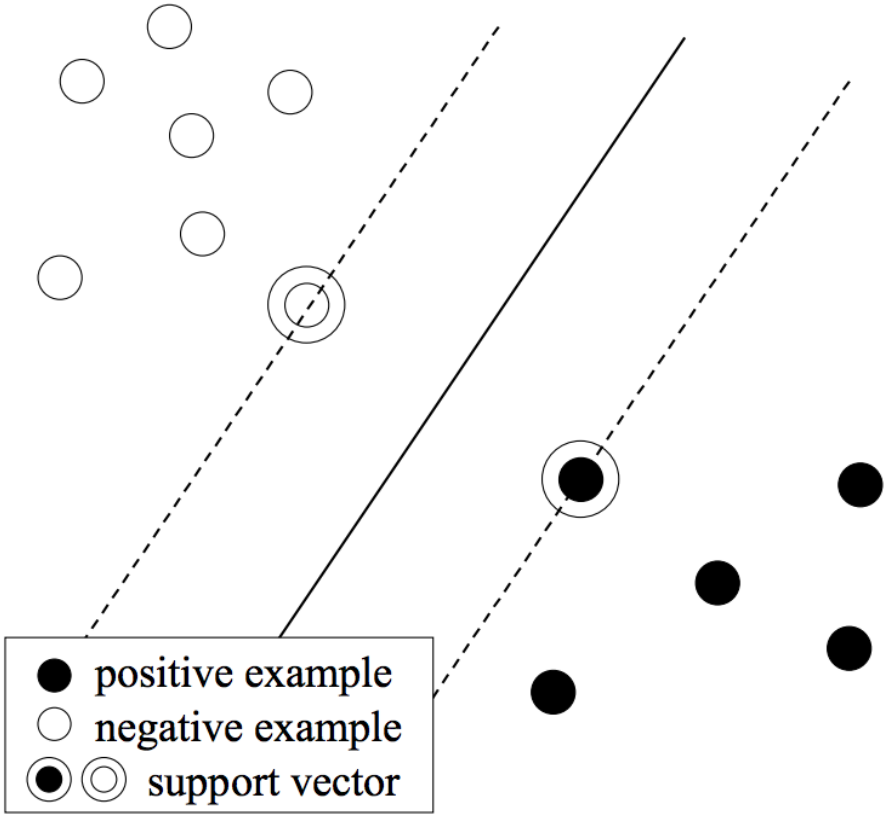



Рисунок 3.2 – Алгоритм опорних векторів

### 3.4.4 Обґрунтування алгоритму дерева ухвалення рішень

У роботі [7] наведено детальний опис роботи цього алгоритму. Розглянемо його основні положення. Для побудови дерева прийняття рішень необхідно знайти індекс ентропії, який обчислюється за формулами:

$$E(D) = \sum_{i=1}^{n} p_i \log(p_i), \qquad (3.26)$$

$$p_i = \frac{|D_{c_i}|}{D}, \qquad (3.27)$$

де $D_{c_i}$ – підмножина дата сету, у якій цільове значення дорівнює конкретній категорії $c_i$ .

Для того, щоб визначити на якому критерії (слові) будувати нову гілку треба використовувати функцію G (D, A), яка перебираючи кожен із елементів (слів) вибірки вирішує чи треба на конкретному слові робити розгалуження.

$$G(D,A) = E(D) - \sum_{v \in V(A)} \frac{|D_{A=v}|}{|D|} E(D_{A=v}), \qquad (3.28)$$

де $E(D)$ – індекс ентропії для дата сету D, $V(A)$ – множина всіх значень із конкретного атрибуту.

Таким чином будується дерево ухвалення рішень на основі якого класифікують тексти.

Зауважимо, що необхідності у тому, аби програмувати дані методи немає, адже у мові програмування Python, яка використовувалася для написання програмного продукту передбачено бібліотеки, що містять функції для створення моделей машинного навчання. Детальну інформацію про ці бібліотеки та функції наведено у розділі 5 дипломного проекту.

### 3.5 Приклад розв'язання задачі

Повернемося до задачі поставленої у розділі 3.1 «Змістовна постановка задачі». Необхідно проаналізувати відгуки та визначити чи є конкретний відгук позитивним або негативним.

На першому етапі використовуватимемо методи обробки природньої мови. В результаті цього етапу отримали корпус слів, зображений на рисунку 3.3

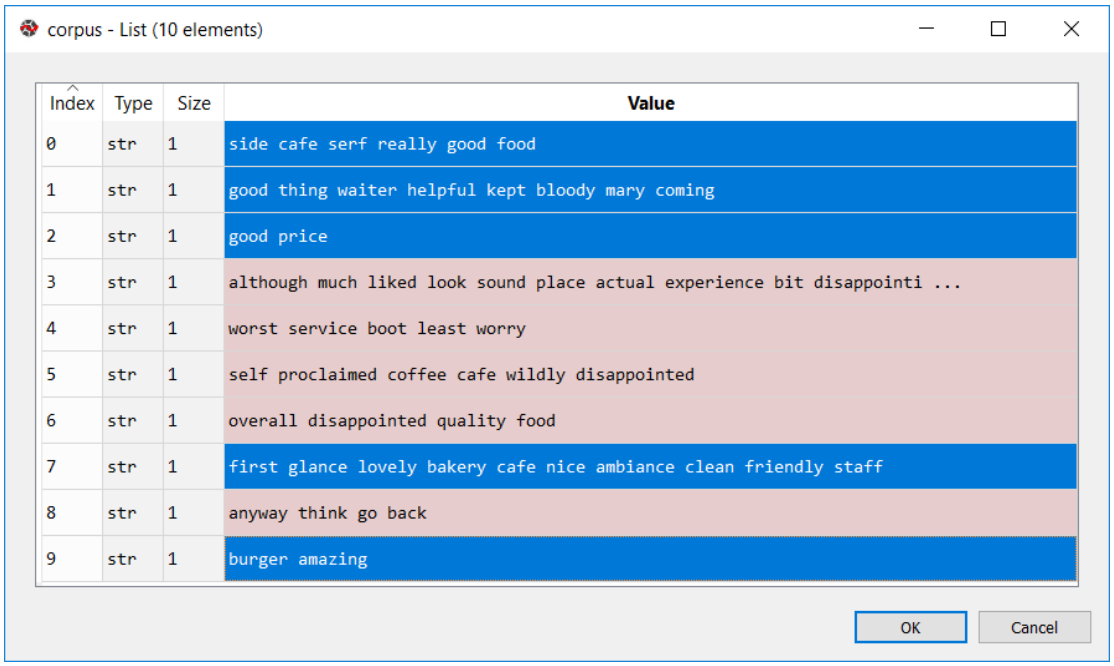



Рисунок 3.3 – Корпус слів

Нехай потрібно визначити до якої категорії належить відгук «Everything was good and tasty!». Після етапу обробки природньої мови цей текст набуде вигляду: «everything good tasty»

Для початку порахуємо ймовірність того, що текст належатиме до категорії «Позитивний відгук» та ймовірність того, що текст : Всього текстів 10. Позитивних відгуків 5, негативних відгуків 5:

$$P_{поз} = \frac{5}{10} = \frac{1}{2},$$

$$P_{нег} = \frac{5}{10} = \frac{1}{2}.$$

Тепер для кожної категорії необхідно побудувати множину слів V і для кожного слова із обчислити частоту його появи у цій множині:

$$P(\text{‘everything’}|\ \text{‘Негативний відгук’}) = \frac{0+1}{29},$$

$$P(\text{‘everything’}|\ \text{‘Позитивний відгук’}) = \frac{0+1}{28},$$

$$P(\text{‘good’}|\ \text{‘Негативний відгук’}) = \frac{0+1}{29},$$

P(‘good’| ‘Позитивний відгук’) = $\frac{3+1}{28}$,

P(‘tasty’| ‘Негативний відгук’) = $\frac{0+1}{29}$,

P(‘tasty’| ‘Позитивний відгук’) = $\frac{0+1}{28}$.

Тепер ми маємо все необхідне аби порахувати ймовірності приналежності конкретного тексту до категорій:

$$P('everything\_good\_tasty'|Позитивний\_відгук') \times P_{поз} = \frac{1}{2} \times \frac{1 \times 5 \times 1}{28 \times 28 \times 28} = 1.1388 \times 10^{-4}$$

та

$$P('everything\_good\_tasty'|Негативний\_відгук') \times P_{нег} = \frac{1}{2} \times \frac{1 \times 1 \times 1}{29 \times 29 \times 29} = 2.05 \times 10^{-5}$$

Оскільки ймовірність того, що розглянутий відгук відноситься до категорії «Позитивний відгук» вища, то робимо висновок, що текст належить цій категорії.

**Висновок до розділу**

Під час написання даного розділу дипломного проекту було наведено змістовну та математичну постановки задачі, визначено основні терміни, які будуть використовуватися.

В змістовній постановці задачі було наведено приклад, який допоможе детальніше ознайомитися із методами обробки природної мови та машинного навчання.

В математичні постановці задачі було формалізовано саму задачі та наведено її основні атрибути. Визначено математичний запис задачі.

Також, було наведено опис методів розв’язання задачі, наведено коротку характеристику кожного алгоритму. Встановлено переваги та недоліки алгоритмів.

Окрім цього було наведено математичне обґрунтування кожного із методів. Встановлено, що необхідності у тому, аби програмувати дані методи немає, адже у мові програмування Python, яка використовувалася для написання програмного продукту передбачено бібліотеки, що містять функції для створення моделей машинного навчання.

Задача, що була представлена у змістовні постановці була розв’язана одним із алгоритмів для ознайомлення із принципами роботи методів обробки природної мови та машинного навчання.

# 4 ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

## 4.1 Засоби розробки

Для реалізації програмного продукту було проаналізовано різні мови програмування, щоб визначити яка з них найкраще підійде для вирішення поставлених задач та досягнення мети. Вирішено розробляти програмний продукт за допомогою мови програмування Python.

Python – приклад високорівневої мови програмування. Python вважається інтерпретованою мовою, оскільки програми Python виконуються інтерпретатором. Існує два способи використання інтерпретатора: інтерактивний режим і режим скрипту. У інтерактивному режимі розробник вводить код програми Python, а інтерпретатор відображає результат. Крім того, розробник може зберігати код у файлі та використовувати інтерпретатор для виконання вмісту файлу, який називається скриптом [10].

Варто зазначити, що Python має можливість використовувати безліч бібліотек, що спрощує роботу розробнику, залишаючи за ним можливість збирати програмний продукт із вже готових блоків.

Мова програмування Python є дуже зручною для вирішення задач подібного типу, адже, як було зазначено раніше, містить багато бібліотек, що спрощують механізми проведення складних обчислень, необхідних для побудови моделей машинного навчання, а також надають доступ до багатьох функцій, що допоможуть у вирішенні поставлених задач.

У розділі 3.4 дипломного проекту «Обґрунтування методів розв’язання» наводилися математичні мети розв’язання поставленої задачі, які необхідно використовувати для побудови моделей машинного навчання. Варто зазначити, що всі ці методи, хоч і вважаються сучасними, але були запропоновані ще у кінці 90-х років минулого сторіччя, тому необхідності

витрачати час на їх реалізацію немає, адже до всіх цих методів є відкритий доступ через середовище розробки мови програмування Python.

Розглянемо, які саме бібліотеки були використані під час розробки програмного продукту та наведемо їх коротку характеристику.

Умовно ці бібліотеки можна поділити на 3 типи: бібліотеки, які використовуються для проходження етапів природної обробки мови, створення моделей машинного навчання та створення веб-застосування.

Детальну характеристику цих бібліотек наведено у таблиці 4.1.

Таблиця 4.1 – Бібліотеки, що були використані під час розробки програмного продукту

| **Назва бібліотеки** | **Опис** |
|---|---|
| re | За допомогою цієї бібліотеки можна прибирати символи із текстів та перетворювати символи із верхнього у нижній регістр. |
| nltk | Ця бібліотека містить розширення stopwords, за допомогою якого із тексту можна прибирати зайві слова, такі як «the», «in», «here», тощо. |
| sklearn | Дана бібліотека містить безліч функцій для створення, тестування та оцінки моделей машинного навчання. Окрім цього, за допомогою цієї бібліотеки, а саме її методу CountVectorizer(), можна представляти текстові файли, у вигляді таблиць для подальшого створення моделей. |
| pandas | Бібліотека, що дає можливість зручно обробляти набори текстів – дата сети |
| numpy | Бібліотека, що дає можливість працювати із числовими структурами даних. |
| Flask | За допомогою цієї бібліотеки можна представляти програмний продукт у вигляді веб-застосунку а також представляти комп’ютер що використовується для запуску програми у вигляді серверу. |
| Bootstrap | Ця бібліотека допомагає поєднувати програмний код і веб-сторінки. Є доповненням до бібліотеки Flask. |

Окрім мови програмування Python використовувалися й інші засоби розробки:

- мова гіпертекстової розмітки документів HTML;
- мова для опису зовнішнього вигляду веб-сторінок CSS;
- фреймворк Bootstrap.

Окрім цього для реалізації програмного продукту використовувалася система управління базами даних PostgreSQL.

PostgreSQL є об'єктно-реляційною системою управління базами даних, розробленою в Каліфорнійському університеті в відділі комп'ютерних наук Берклі. Компанія POSTGRES впровадила багато концепцій, які стали доступними лише в деяких комерційних системах баз даних значно пізніше. PostgreSQL містить відкритий вихідний код, що підтримує велику частину стандартів SQL і пропонує багато сучасних можливостей, таких як складні запити, зовнішні ключі, тригери, оновлення переглядів, цілісність транзакцій, контроль багатоваріантного паралелізму, тощо. Крім того, PostgreSQL може бути розширена користувачем багатьма способами, наприклад, шляхом додавання нових типів даних, функції, операторів, сукупних функції, індексних методів, тощо. Через ліберальну ліцензію, PostgreSQL можна використовувати, змінювати [11].

### 4.2 Загальні вимоги до технічного забезпечення

Даний програмний продукт представлений у вигляді веб-застосування і складається із серверної та клієнтської частин.

Для коректної роботи серверної частини веб застосування необхідно використовувати сервер що може бути представлений локальною обчислювальною машиною. Сервер повинен мати такі конфігурації:

- процесор з тактовою частотою не нижче 2 ГГц;
- достатній об’єм оперативної пам’яті (не менше 2 ГБ);

- жорсткий диск (не менше 40 ГБ);
- операційна система Windows 7 і вище;
- база даних PostgreSQL 10.4 і вище;
- Python 3.5.

Для коректної роботи клієнтської частини веб застосування необхідно:

- підключення до мережі Інтернет;
- дата сет для класифікації текстів.

Планується першу версію програмного забезпечення розробити для випадку, коли серверною частиною виступає локальна обчислювальна машина, на якій запускається програмний продукт. У цьому випадку для коректної роботи клієнтської частини застосування, локальна обчислювальна машина повинна мати такі параметри:

- процесор з тактовою частотою не нижче 2 ГГц;
- достатній об’єм оперативної пам’яті (не менше 2 ГБ);
- жорсткий диск (не менше 40 ГБ);
- операційна система Windows 7 і вище;
- база даних PostgreSQL 10.4 і вище;
- Python 3.5.

Система повинна адекватно реагувати на помилки серверної частини застосування та видавати відповідні повідомленні користувачеві, а також записувати їх у звіт для адміністратора. Цим буде забезпечена можливість подальшого відналагодження системи.

Для адекватної роботи системи необхідне забезпечено безперебійне живлення на серверній частині веб-застосування та підключення до мережі Інтернет зі сторони Користувачів системи.

Розміщення обладнання, технічних засобів повинно відповідати вимогам техніки безпеки.

Усі користувачі системи повинні дотримуватися правил експлуатації електронної обчислювальної техніки.

### 4.3 Архітектура програмного забезпечення

#### 4.3.1 Схема архітектури програмного забезпечення

Розглянемо схему структурну архітектури програмного забезпечення, що представлена у розділі графічного матеріалу.

Архітектура програмного забезпечення складається із чотирьох рівнів: рівень представлення, рівень бізнес логіки, рівень доступу до даних та рівень даних.

На рівні представлення знаходяться такі компоненти: User Interface: Web Application (інтерфейс користувача, що представлений веб-застосунком), HTML (мова гіпертекстової розмітки документів) CSS(каскадні таблиці стилів – мова опису зовнішнього вигляду веб-сторінок).

Рівень бізнес логіки складається із таких компонентів: Front-end Framework: Bootstarp (інтерфейс взаємодії користувача веб-застосунку та його функціями), Functions (функції, що виконуються в рамках системи), Back-end: Python 3.5 (опис функціоналу системи, що представлений мовою програмування Python версії 3.5) та Web-Framework: Flask (веб-інтерфейс взаємодії інтерфейсу користувача та функції системи).

Опис функціоналу системи в свою чергу представлений бібліотеками мови програмування Python версії 3.5, а саме:

- pandas, numpy (бібліотеки для роботи із дата сетами);
- psycopg2 (бібліотека для роботи із сервером баз даних);
- nltk (бібліотека для роботи із функціями обробки природної мови);
- sckit-learn (бібліотека для роботи із функціями машинного навчання).

Функції веб-застосунку можна поділити на Built-In Web Fuctions (вбудовані функції) та Extended Web-Functions (функції реалізовані в рамках створення системи).

Вбудованими функіями є:

- Loading the file (завантаження файлу);
- Form validating (валідація форм);
- Redirection functions (функції перенаправлення між сторінками).

Функціями реалізованими в системі є:

- Natural language processing methods (методи обробки природної мови);
- Machine learning methods (методи машинного навчання);
- Metric functions (функції оцінки моделей машинного навчання);
- Prediction dunctions (функції класифікації тестів).

На рівні доступу до даних реалізовані функції за допомогою яких можна отримувати інформацію із серверу баз даних та інформацію про локальні файли, такі як зображення, тестові файли, файли із дата сетами та моделями машинного навчання.

Головним керуючим елементом даного програмного забезпечення є Сервер веб-застосування. Саме він забезпечує коректну комунікацію між сторінками веб-застосунку, а також надає доступ до серверу баз даних. Сервер баз даних містить основну інформацію про згенеровані задачі. Карта веб-застосунку представлена на рисунку 4.1.

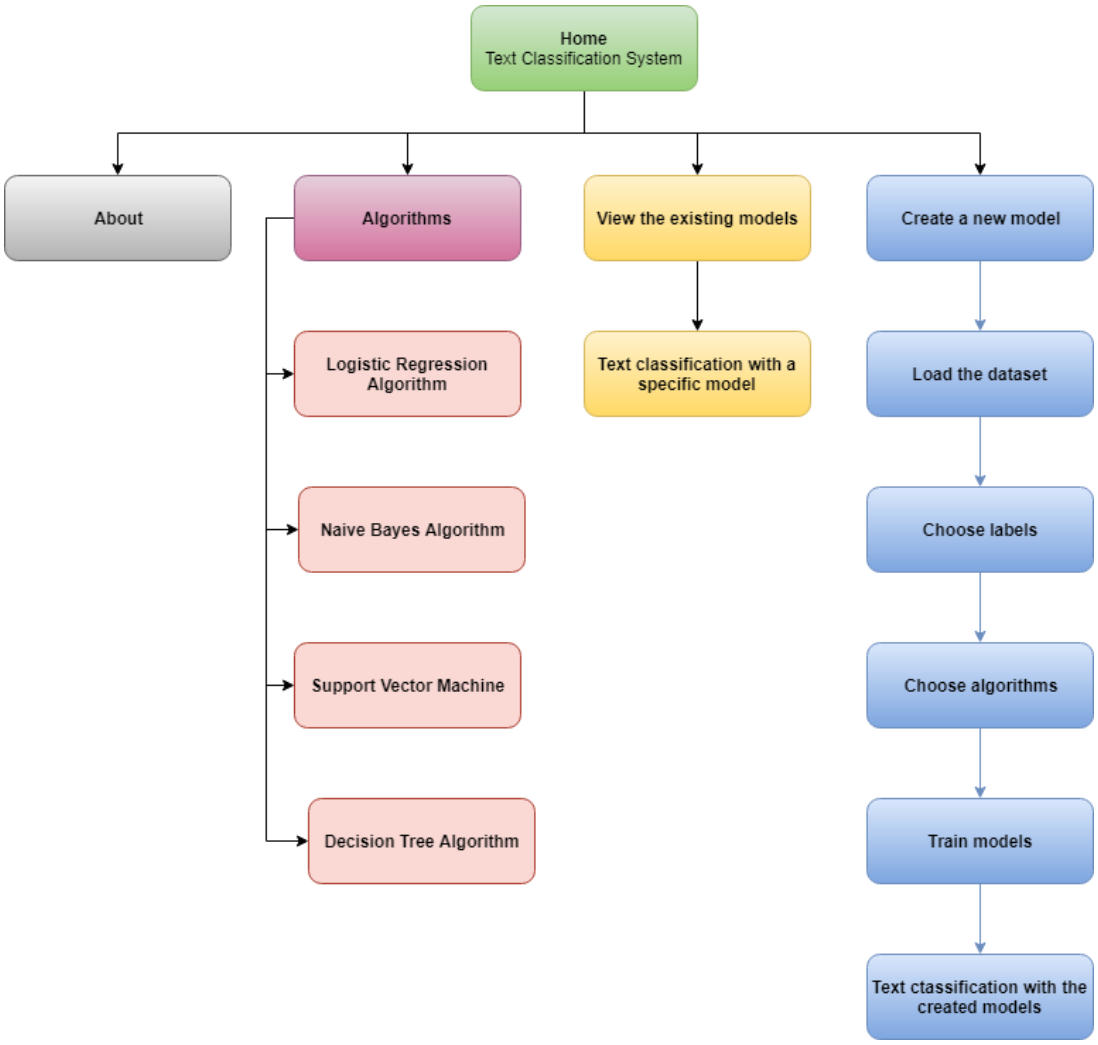



Рисунок 4.1 – Схема структурна карти застосунку.

Веб-застосування складається з 15 веб-сторінок: Home (Домашня сторінка), About (Інформація про систему), Algorithms (Алгоритми), Logistic Regression Algorithm (Алгоритм логістичної регресії), Naive Bayes Algorithm (Алгоритм наївного Байєса), Support Vector Machine (Алгоритм опорних векторів), Decision Tree Algorithm (Алгоритм дерева прийняття рішень), View the existing models (Переглянути створені моделі), Text classification with a specific model (Класифікація текстів за допомогою конкретної моделі), Create a new model (Створити нову модель), Load the dataset (Завантажити дата сет), Choose labels (Вибрати мітки), Choose algorithms (Вибрати алгоритми), Train models (Навчити моделі), Text classification with the created models

(Класифікація текстів за допомогою створених моделей). Схема структурна архітектури програмного забезпечення представлена у розділі графічного матеріалу.

На сторінці Home (Домашня сторінка) користувач повинен мати змогу перейти на сторінку About (Інформація про систему), де представлена загальна інформація про систему та її автора, перейти на сторінку Algorithms (Алгоритми), де користувач має змогу ознайомитися із загальною теорію про алгоритми машинного навчання, перейти на сторінку View the existing models (Переглянути створені моделі), де користувач може переглянути загальну інформацію про моделі машинного навчання, що були створені раніше та перейти на сторінку Create a new model (Створити нову модель), де користувач може створити власні моделі машинного навчання на основі дата сету.

На сторінці About (Інформація про систему) користувач може ознайомитися із загальною інформацію про систему класифікації текстів та її авторів.

На сторінці Algorithms (Алгоритми) користувач може ознайомитися із загальною теорію про алгоритми машинного навчання та перейти на сторінку Logistic Regression Algorithm (Алгоритм логістичної регресії), Naïve Bayes Algorithm (Алгоритм наївного Байєса), Support Vector Machine (Алгоритм опорних векторів) або Decision Tree Algorithm (Алгоритм дерева прийняття рішень).

На сторінці Logistic Regression Algorithm (Алгоритм логістичної регресії) користувач має змогу ознайомитися із основними положеннями про алгоритм логістичної регресії.

На сторінці Naive Bayes Algorithm (Алгоритм наївного Байєса) користувач має змогу ознайомитися із основними положеннями про алгоритм наївного Байєса.

На сторінці Support Vector Machine (Алгоритм опорних векторів) користувач має змогу ознайомитися із основними положеннями про алгоритм опорних векторів.

На сторінці Decision Tree Algorithm (Алгоритм дерева прийняття рішень) користувач має змогу ознайомитися із основними положеннями про алгоритм дерева прийняття рішень.

На сторінці View the existing models (Переглянути створені моделі) користувач має змогу під’єднатися до таблиці бази даних в якій містяться записи про створені моделі машинного навчання. Окрім цього, при виборі певної моделі користувач переходить на сторінку Text classification with a specific model (Класифікація текстів за допомогою конкретної моделі) де має змогу класифікувати конкретний текст за допомогою обраної моделі до однієї із доступних категорій.

На сторінці Create a new model (Створити нову модель) користувач розпочинає процес створення моделей машинного навчання, що автоматично переходить на сторінку Load the dataset (Завантажити дата сет), де користувачеві пропонується завантажити дата сет у систему. На наступному кроці користувач має обрати колонку із дата сету, де розміщені тексти та колонку, де розміщені категорії текстів. Це відбувається на сторінці Choose labels (Вибрати мітки). На наступному кроці користувачеві пропонується обрати алгоритми за допомогою яких він хоче створити моделі машинного навчання на сторінці Choose algorithms (Вибрати алгоритми). На наступному кроці відбувається процес навчання моделей на сторінці Train models (Навчити моделі), після якого користувачеві пропонується детальна інформація про створені моделі. За бажанням користувач може перейти на сторінку Text classification with the created models (Класифікація текстів за допомогою створених моделей) для класифікації конкретних текстів та безпосереднього використання щойно створених моделей.

### 4.3.2 Діаграма класів

Розроблений програмний продукт складається з чотирьох класів: клас Текст (Text), клас Дата Сет (Dataset), клас Модель (Model) та клас Користувач (User).

Клас Текст (Text) містить такі атрибути:

- ім’я дата сету (dataset_name), що має тип рядка (string);
- індекс у дата сеті (dataset_index), що має тип цілочислового значення (int).

Клас Текст (Text) містить такі методи:

- прибрати зайві символи (re.sub()), що повертає тип рядка (string);
- перетворити символи у нижній регістр (lower()), що має тип рядка (string);
- прибрати не значимі слова із тексту (wnl.lemmatize()), що має тип рядка (string).

Клас Дата Сет (Dataset) містить такі атрибути:

- шлях до файлу (path_to_the_file), що має тип рядка (string);
- шлях до корпусу слів (path_to_the_corpus), що має тип рядка (string);
- ім’я (name), що має тип рядка (string).

Клас Дата Сет (Dataset) містить такий метод: створити корпус слів (CreateCorpus()), що повертає тип масиву (array);

Клас Модель (Model) містить такі атрибути:

- ім’я (name), що має тип рядка (string);
- точність (accuracy), що має тип числа з плаваючою комою (double);
- розмір тренувальної вибірки (train_size), що має тип числа з плаваючою комою (double);
- кількість ознак (max_features), що має тип цілочислового значення (int).

Клас Модель (Model) містить такі методи:

- тренувати модель (Train());
- тестувати модель (Test());
- класифікувати текст (Predict()), що повертає тип рядка (string);
- відображати результат (ShowResults()), що повертає тип рядка (string).

Клас Користувач (User) містить такі методи:

- створити модель (CreateModel());
- завантажити дата сет (LoadDataset());
- визначити мітки (DefineLabels())
- обрати алгоритми (ChooseAlgorithms())
- класифікувати текст (ClassifyTexts()).

Опишемо зв'язки між класами. Клас Дата Сет узагальнює клас Текст (Text). В одному дата сеті міститься багато текстів. Клас Дата Сет (Dataset) та клас Модель (Model) поєднані n-арною асоціацією, тобто на основі одного дата сету може бути створено багато моделей машинного навчання. Клас Дата Сет (Dataset) та клас Модель (Model) поєднані n-арною асоціацією, тобто один користувач може створювати безліч моделей машинного навчання.

Схема структурна класів програмного забезпечення наведена у розділі графічного матеріалу.

### 4.3.3 Діаграма послідовності

Розглянемо схему структурну послідовності, представлену у розділі графічного матеріалу. Користувач має можливість створити модель машинного навчання. Для цього необхідно використати функцію CreateModel(). У разі успішного виконання цієї функції користувач може завантажити дата сет в систему за допомогою функції LoadDataset(). У разі успішного завантаження дата сету в систему користувачеві пропонується вказати в якому стовбці знаходяться тексти і в якому – їх категорії. Після

того, як мітки були визначені, створюється корпус слів за допомогою функції CreateCorpus(). Для створення корпусу слів необхідно виконати функції класу Text, а саме: прибрати зайві символи (re.sub()), перетворити символи у нижній регістр (lower()) та прибрати незначимі слова із тексту (wnl.lemmatize()). Після успішного створення корпусу слів користувачеві надається можливість обрати алгоритми на основі яких створюватимуться моделі машинного навчання. Варто зазначити, що на основі одного алгоритму та дата сету може бути створена лише одна модель. Після того як користувач обрав алгоритми за допомогою функції ChooseAlgorithms(), виконується навчання, тестування та оцінка моделей машинного навчання за допомогою функцій Train(), Test() та Evaluate() відповідно. Далі за допомогою функції ShowResults(), виводиться детальна інформація про створену модель. Якщо користувач хоче класифікувати конкретний тест, то необхідно використати функцію Classify(), яка в свою чергу викличе функцію Predict() для визначення категорії до якої відноситься той чи інший текст. Результати будуть виведені за допомогою функції ShowResults().

### 4.3.4 Діаграма компонентів

Розглянемо схему структурну компонентів, представлену у розділі графічного матеріалу. З діаграми видн, що система містить такі компоненти: Алгоритм (Algorithm), Модель (Model), Користувач (User), Дата Сет (Dataset), Веб Застосування (Web Application) та База Даних (Database).

Компонент Алгоритм (Algorithm) надає компоненту Модель (Model) необхідну інформацію про своє функціонування.

Компонент Дата Сет (Dataset) надає компоненту Модель (Model) інформацію про тексти та їх категорії.

Компонент Модель (Model) надає компоненту Веб Застосування (Web Application) інформацію про створену на основі конкретного алгоритму та конкретного дата сету модель.

Компонент Користувач (User) надає компоненту Веб Застосування (Web Application) інформацію про те, яку саме модель він хоче створити.

Компонент База Даних (Database) надає компоненту Веб Застосування (Web Application) інформацію про створені моделі у разі якщо користувач хоче їх переглянути.

### 4.3.5 Специфікація функцій

Опишемо функції класів програмного забезпечення. Детальний опис усіх функцій наведено у таблиці 4.2

Таблиця 4.2 – Специфікація функцій програмного забезпечення

| Назва функції | Опис |
|---|---|
| re.sub(string) | Функція описана у бібліотеці re мови програмування Python. Функція прибирає розділові знаки із текстів. |
| lower(string) | Функція перетворює всі символи до нижнього регістру. |
| wnl.lemmatize(string) | Функція описана у бібліотеці wnl мови програмування Python. Функція прибирає всі незначимі слова із текстів. |
| CreateModel() | Функція класу User (Користувач), що призначена для створення моделей машинного навчання. |
| LoadDataset() | Функція класу User (Користувач), що призначена для завантаження дата сету в систему. |
| DefineLabels() | Функція класу User (Користувач), що призначена для визначення стовпця із текстами та стовпця із категоріями. |
| ChooseAlgorithms() | Функція класу User (Користувач), що призначена для вибору алгоритмів на основі яких будуть створені моделі машинного навчання. |
| ClassifyText() | Функція класу User (Користувач), що призначена для класифікації текстів. |

Продовження таблиці 4.2

| Назва функції | Опис |
|---|---|
| CreateCorpus() | Функція класу Dataset (Дата Сет), що призначена для створення корпусу слів – числового представлення дата сету. |
| Train() | Функція класу Model (Модель), що призначена для тренування моделей машинного навчання. |
| Test() | Функція класу Model (Модель), що призначена для тестування моделей машинного навчання. |
| Predict() | Функція класу Model (Модель), що призначена для передбачення категорії до якої відноситься той чи інший текст. |
| ShowResults() | Функція класу Model (Модель), що призначена для виведення інформації про |

### 4.4 Опис звітів

Під час тестування моделей машинного навчання, генерується матриця невідповідностей (confusion matrix). Саме вона дає достатню повну інформацію про ефективність конкретної моделі машинного навчання. Розглянемо приклад побудови матриці невідповідностей та покажемо, як виходячи із цієї матриці можна порахувати точність створеної моделі.

Нехай в ході виконання програми тестова вибірка складала 100 текстів, що були класифіковані на 3 категорії. Тоді матриця невідповідностей матиме вигляд, представлений у таблиці 4.3.

Таблиця 4.3 – Структура матриці невідповідностей

| | Кількість текстів, що насправді належать категорії 1 | Кількість текстів, що насправді належать категорії 2 | Кількість текстів, що насправді належать категорії 3 |
|---|---|---|---|
| Кількість текстів, що біли віднесені побудованою моделлю машинного навчання до категорії 1 | 33 | 3 | 1 |
| Кількість текстів, що біли віднесені побудованою моделлю машинного навчання до категорії 2 | 4 | 28 | 2 |
| Кількість текстів, що біли віднесені побудованою моделлю машинного навчання до категорії 3 | 2 | 2 | 25 |

Розглянемо другу колонку із таблиці 4.3. Видно, що всього було 39 текстів із категорії 1. Із них 35 текстів модель машинного навчання віднесла до категорії 1, 4 тексти до категорії 2 та 2 тексти до категорії 3.

Для того аби порахувати точність такої моделі, необхідно кількість правильно класифікованих текстів поділити на загальну кількість текстів та помножити на 100%. У нашому випадку точність складатиме 86%.

Така матриця будуватиметься для кожної побудованої моделі машинного навчання.

Окрім цього, після створення кожної моделі, генеруватиметься файл із детальною інформацією про модель. Приклад такого файлу наведено на рисунку 4.1.

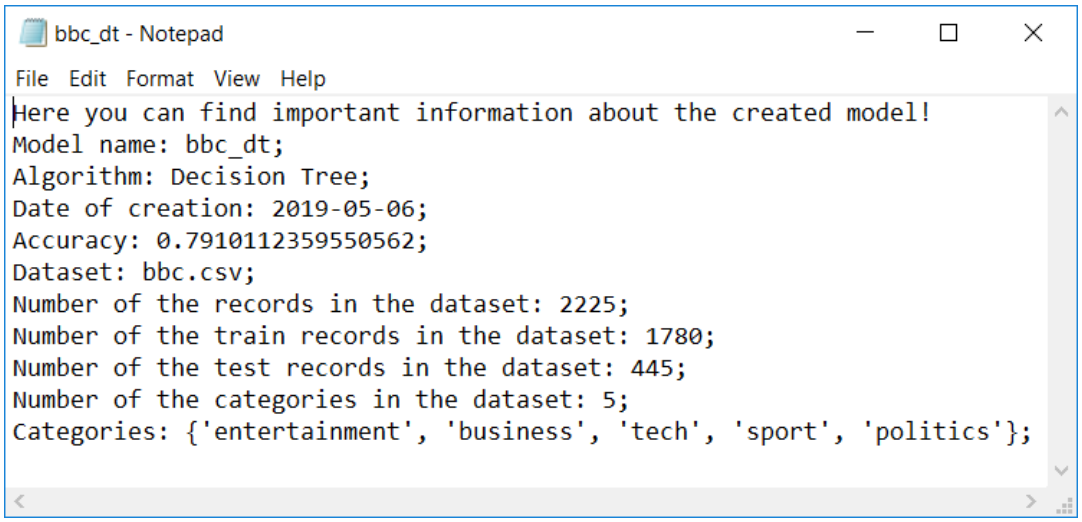



Рисунок 4.1 – Приклад файлу з інформацією про модель

Із рисунку видно, що файл із інформацією про модель машинного навчання містить такі поля: назва моделі, алгоритм, на основі якого була створена модель, дата створення моделі, точність моделі, що обчислюється на основі матриці невідповідностей, назва дата сету, кількість записів у дата сеті, кількість записів із дата сету, що були використані для тренування моделі, кількість записів із дата сету, що були використані для тестування моделі, кількість категорій в дата сеті, список категорій із дата сету.

**Висновок до розділу**

У ході написання даного розділу дипломної роботи було розглянуто засоби розробки програмного забезпечення, наведено коротку характеристику мов програмування, які б могли допомогти у вирішенні поставлених задач та досягненні мети. Обґрунтовано вибір мови програмування Python, а також системи управління базами даних PostgreSQL.

Окрім цього були виявлені загальні вимоги до програмного забезпечення, описано бібліотеки, що використовуватимуться при розробці програмного продукту.

Було описано архітектуру програмного забезпечення. Спроектовано діаграми класів, послідовності та компонентів і наведено їх детальний опис. Також наведено специфікацію функцій, що використовувалися при створенні програмного продукту та описано звіти, що генеруються в результаті роботи програми.

# 5 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

## 5.1 Керівництво користувача

Розглянемо функції користувача, що були реалізовані в системі. Відповідно до розділу 1.1.2 «Опис функціональної системи», користувач має виконувати такі функції в рамках системи:

- завантажувати дата сет в систему;
- ознайомитися із теоретичною частиною алгоритмів машинного навчання;
- обирати алгоритми для створення моделей машинного навчання;
- створювати моделі машинного навчання;
- навчати та тестувати моделі машинного навчання;
- аналізувати ефективності моделей машинного навчання;
- класифікувати тексти.

Під час розробки системи були реалізовані всі функції користувача. Користувач повинен мати можливість обрати у якому режимі працюватиме веб-застосунок: у режимі перегляду створених моделей чи у режимі створення нових моделей машинного навчання. Ця можливість представлена на головній сторінці веб-застосунку, що представлена на рисунку 5.1.

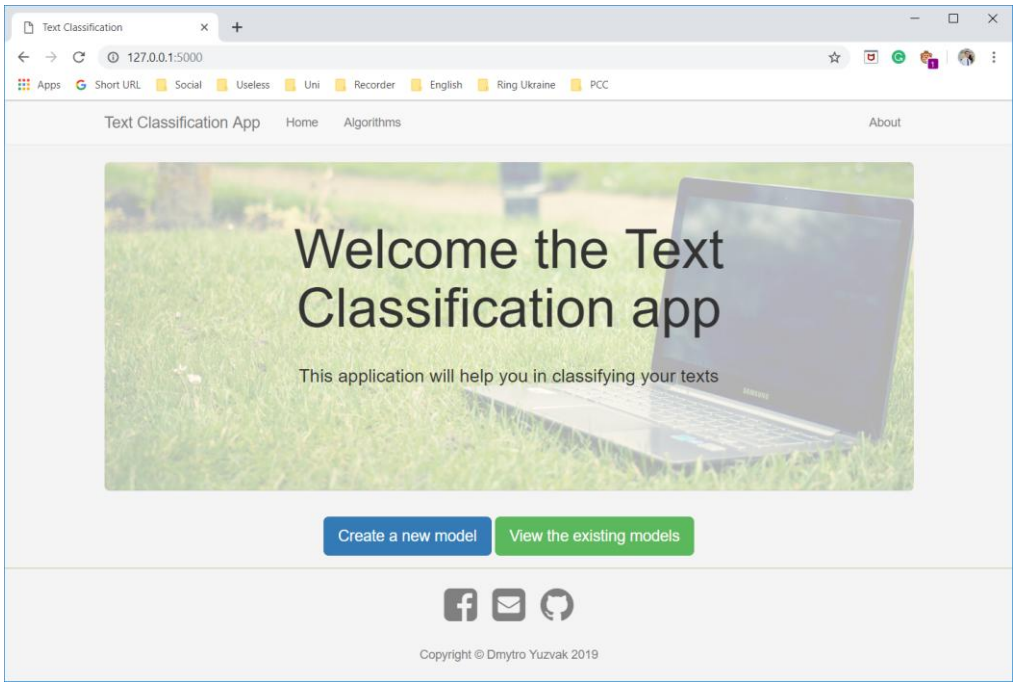



Рисунок 5.1 – Головна сторінка веб-застосунку

Якщо користувач обирає режим створення нової моделі, то йому надається можливість завантажити дата сет.

Розглянемо функцію завантаження дата сету в систему. Користувач повинен мати можливість завантажувати будь-який дата сет, представлений у форматі *.csv, *.tsv або *.xls. Результат роботи функції представлено на рисунку 5.2.

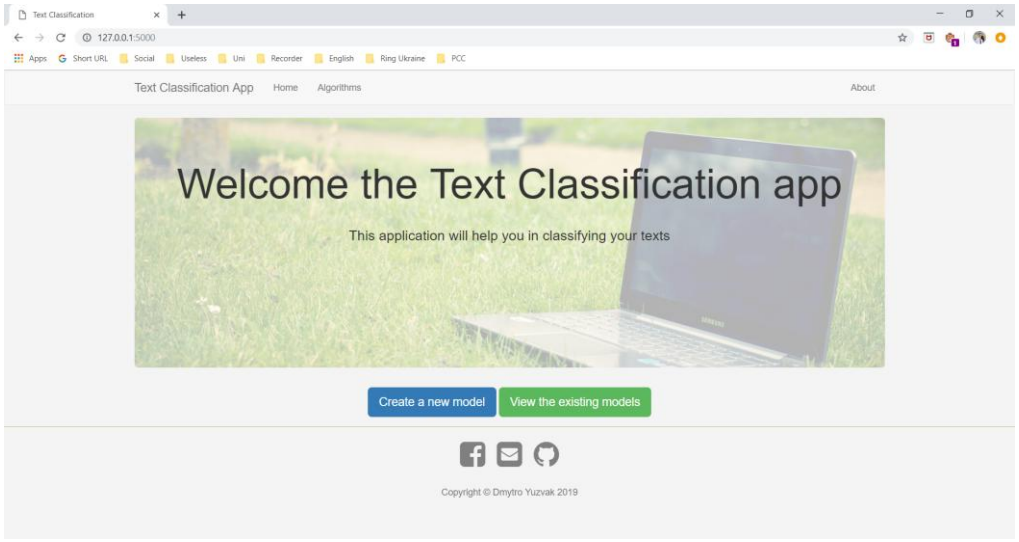



Рисунок 5.2 – Завантаження дата сету в систему.

При натисненні на кнопку Вибрати файл (Choose File) повинно відкритися діалогове вікно для того, аби користувач мав можливість обрати файл дата сету із комп'ютера. Розглянемо рисунок 5.3.

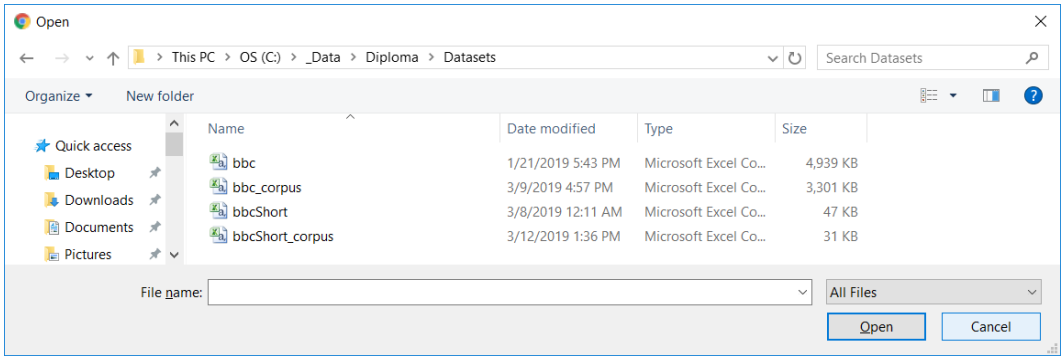



Рисунок 5.3 – Діалогове вікно для завантаження дата сету

Після завантаження дата сету система повинна надати користувачеві можливість переглянути завантажений дата сет та обрати колонку із текстами та категоріями. Результат роботи цих функції представлено на рисунку 5.4

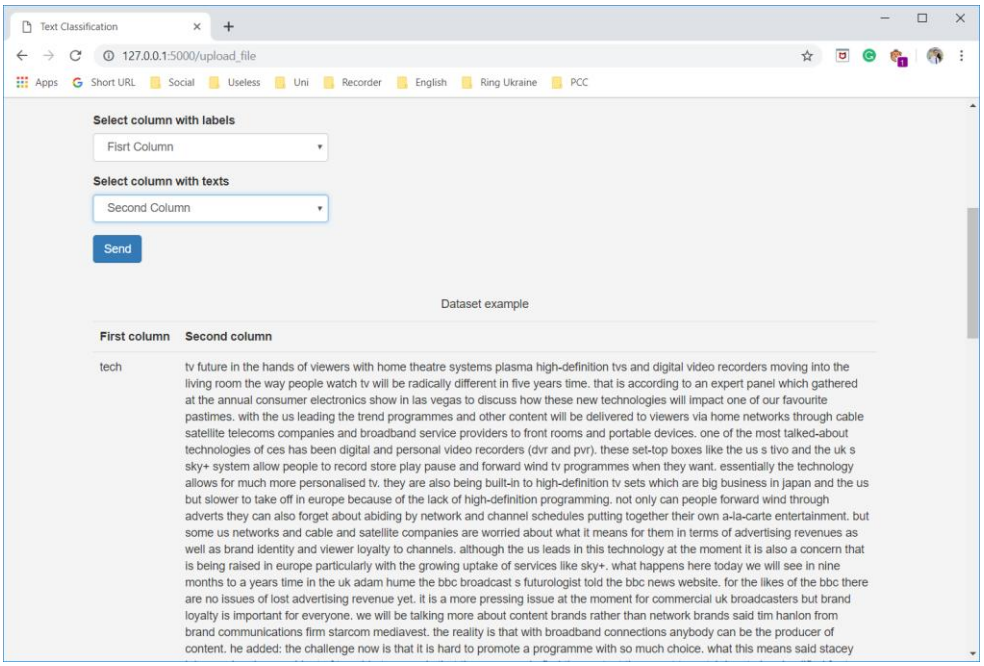



Рисунок 5.4 – Функції перегляду дата сету та вибору вхідних даних

Після того, як користувач вказав вхідні дані, система повинна надати можливість обрати алгоритми для створення моделей машинного навчання.

Реалізація цієї функції представлена на рисунку 5.5.

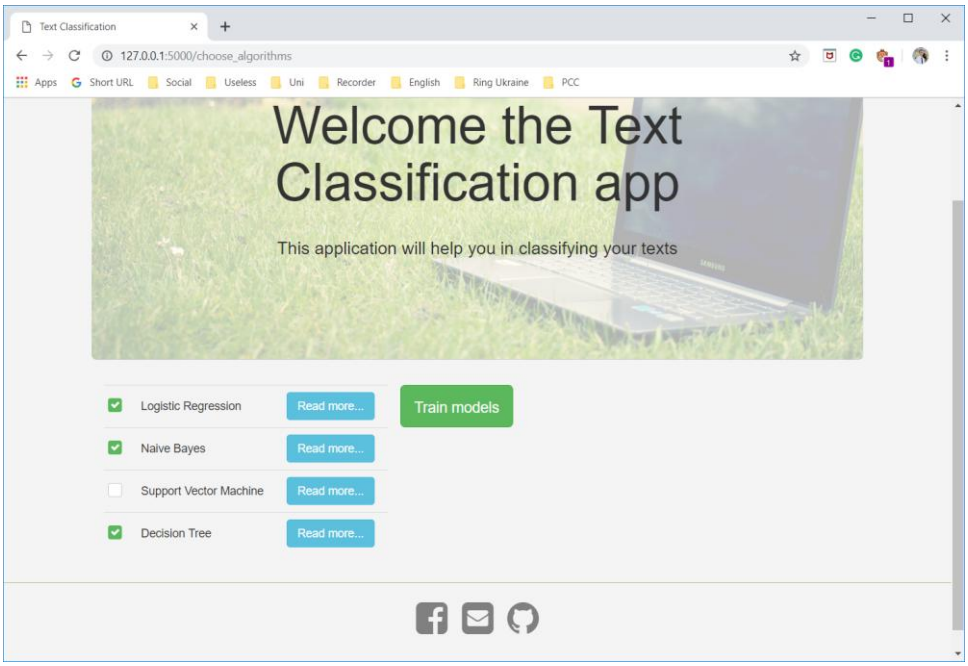



Рисунок 5.5 – Вибір алгоритмів для створення моделей машинного навчання

Також система повинна надати можливість користувачеві переглядати інформацію про алгоритми, що реалізовані в системі. Реалізація функції відображення інформації про алгоритм логістичної регресії представлена на рисунку 5.6. Реалізація функції відображення інформації про алгоритм наївного Байєса представлена на рисунку 5.7. Реалізація функції відображення інформації про алгоритм опорних векторів представлена на рисунку 5.8. Реалізація функції відображення інформації про алгоритм дерева ухвалення рішень представлена на рисунку 5.9.

Рисунок 5.6 – Відображення інформації про алгоритм логістичної регресії




Рисунок 5.7 – Відображення інформації про алгоритм наївного Байєса

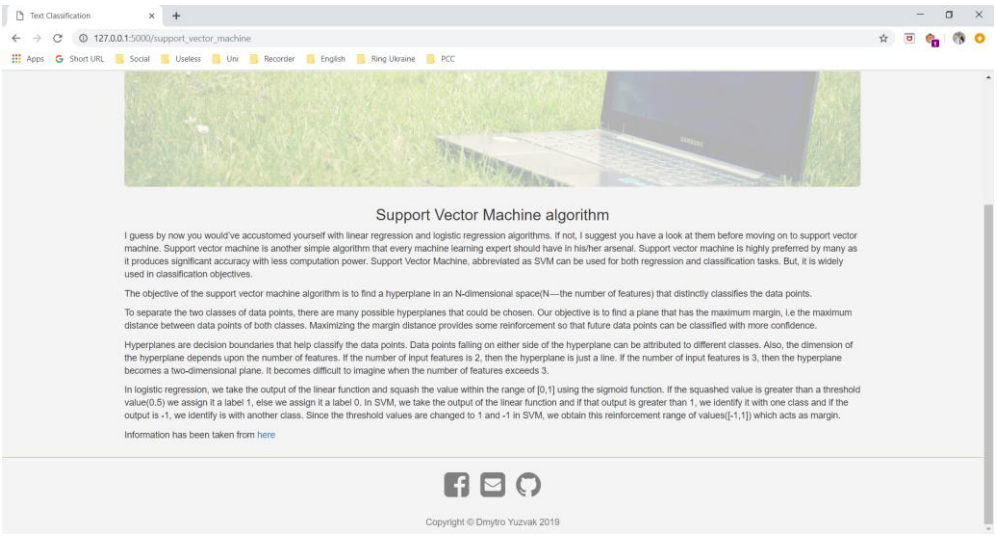



Рисунок 5.8 – Відображення інформації про алгоритм опорних векторів

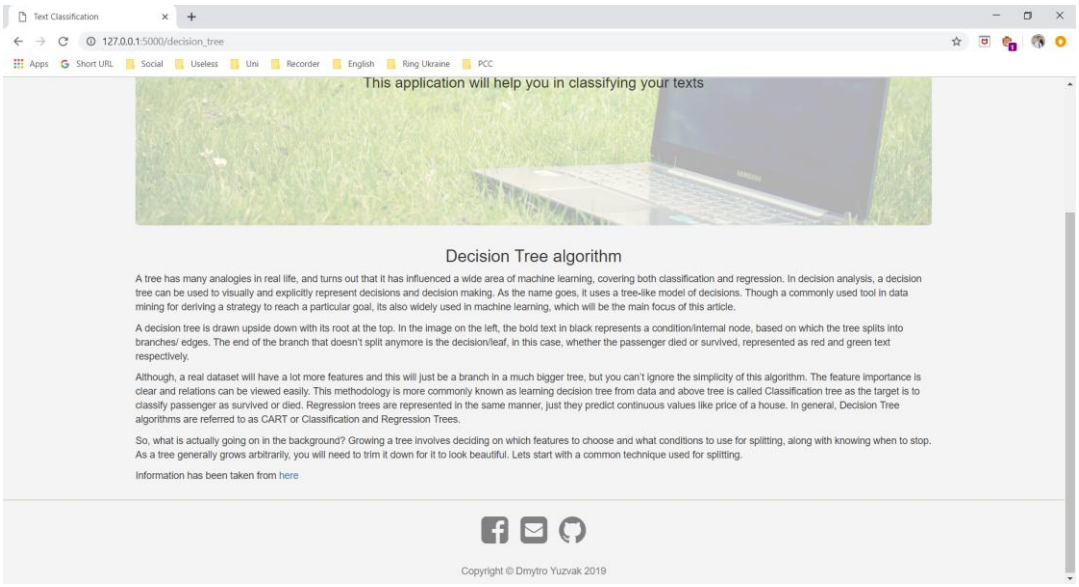



Рисунок 5.9 – Реалізація інформації про алгоритм дерева ухвалення рішень

Після того як користувач обрав алгоритми для створення моделей машинного навчання, система повинна відобразити інформацію про створені моделі та надати можливість класифікувати конкретний текст до певної категорії. Реалізація цієї функції представлена на рисунку 5.10

Рисунок 5.10 – Відображення інформації про створені моделі

Коли користувач обирає модель машинного навчання, якою хоче класифікувати текст, то відкривається нове вікно, у якому система повинна надати можливість користувачеві ввести текст для класифікації. Результат роботи функції зображено на рисунку 5.11.




Рисунок 5.11 – Функція класифікації тексту

Результат роботи цієї функції представлено на рисунку 5.12.

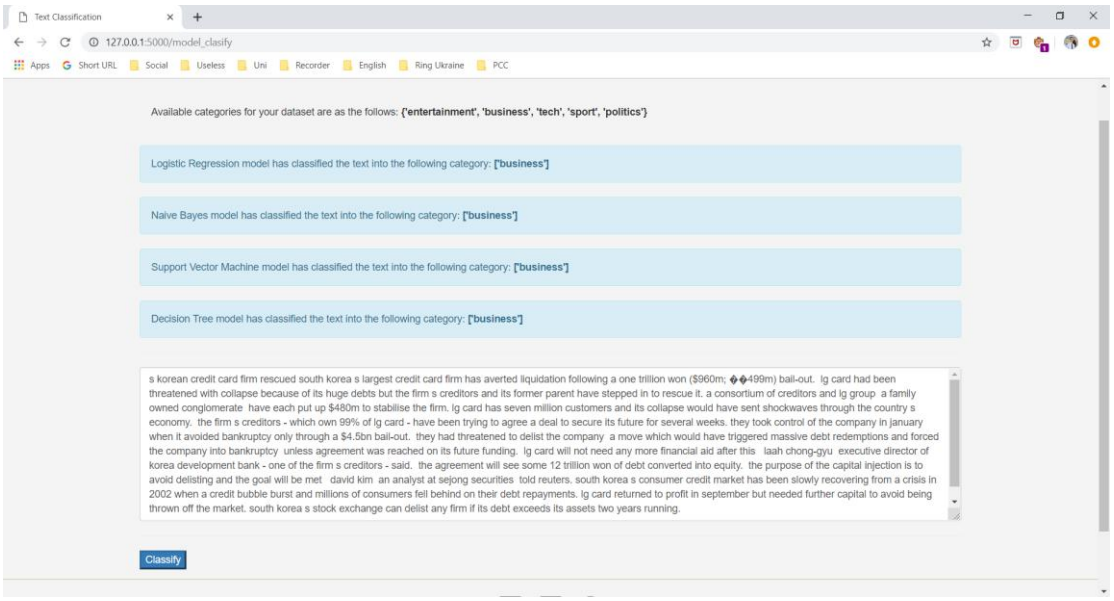



Рисунок 5.12 – Результат роботи функції класифікації тексту.

Користувач повинен мати змогу переглядати раніше створені моделі, інформація про які була записана до бази даних. Результат роботи цієї функції користувача зображено на рисунку 5.13.

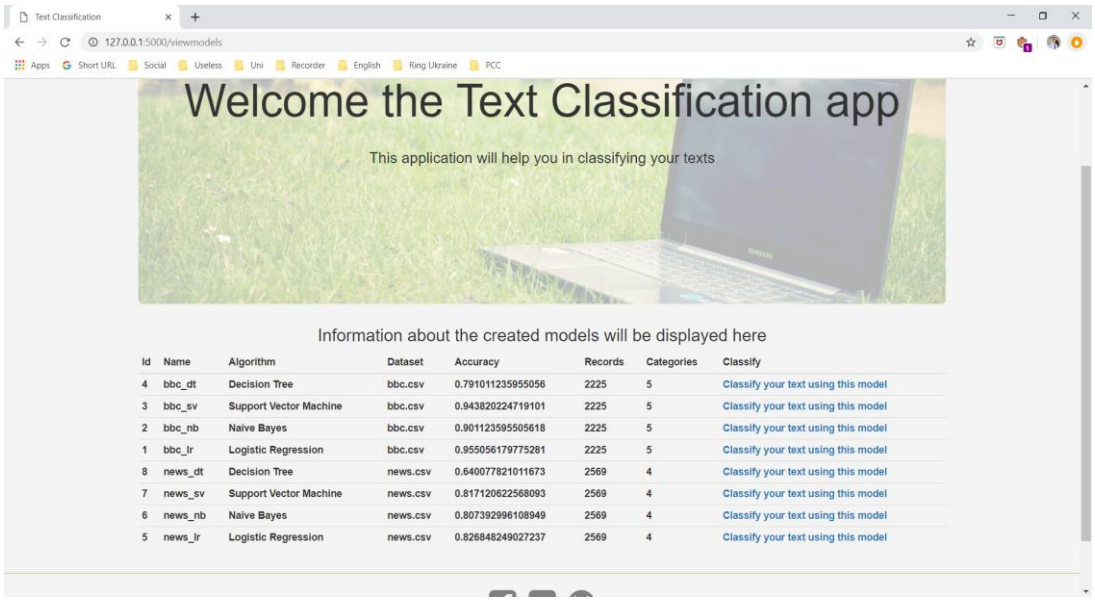



Рисунок 5.13 – Функція перегляду раніше створених моделей

## 5.2 Випробування програмного продукту

### 5.2.1 Мета випробувань

Метою випробувань програмного продукту є виявлення слабких місць у системі та впровадження рішень щодо їх усунення.

Для досягнення даної мети необхідно розглянути функції які виконує система та описати очікуваний результат роботи функцій. На наступному етапі до кожної функції необхідно створити 5-10 тестів. Кілька тестів повинні приймати граничні значення.

Для кожного тесту необхідно виконувати функцію та оцінити достовірність отриманих результатів.

### 5.2.2 Загальні положення

Випробування проводяться на основі таких документів:

– ГОСТ 34.603–92. Інформаційна технологія. Види випробувань автоматизованих систем;

– ГОСТ РД 50-34.698-90. Автоматизовані системи вимог до змісту документів.

### 5.2.3 Результати випробувань

В процесі тестування була перевірена уся функціональність системи. Розглянемо перелік тестових сценаріїв:

- створення нової моделі машинного навчання;
- перегляд створених моделей машинного навчання;
- завантаження дата сеу в систему;
- обрання колонок із текстами та категоріями;
- вибір алгоритмів для створення моделей машинного навчання;
- перегляд інформації про алгоритми;

- створення класифікатора (навчання та тестування);
- оцінка ефективності моделей машинного навчання;
- класифікація текстів за допомогою моделей машинного навчання.

Тести були проведені методом ручного тестування. Всі тести направлені на виявлення невідповідностей системи заявленим характеристикам.

Всі тести були успішно пройдені.

У документі «Програма та методика випробувань» наведений перелік випробувань основних функціональних можливостей.

**Висновок до розділу**

У ході написання даного розділу дипломного проекту, було сформовано керівництво користувача та наведено екранні форми розробленого веб-застосунку.

Було встановлено мету проведення випробувань, наведено загальні положення. Також, визначено тести які необхідно провести над кожною із функцій.

У процесі тестування була перевірена уся функціональність системи, наведений перелік випробувань основних функціональних можливостей.

У ході тестування програмного продукту було встановлено, що функціонал розробленого веб-застосунок відповідає встановленим вимогам. Варто зауважити, що всі функції про які було заявлено в розділі 1.1.2 Опис функціональної моделі були розроблені в рамках системи.

## ЗАГАЛЬНІ ВИСНОВКИ

У сучасному світі більшість інформації представлена у вигляді текстів. Тому зростає необхідність у створенні автоматизованих систем обробки текстових даних. Для того, щоб вирішити конкретну задачу класифікації текстів потрібно витратити багато ресурсів, що не є прийнятим особливо коли розглядаються комплексні задачі. Тому виникає необхідність у створенні універсального методу вирішення поставленої задачі.

На першому етапі роботи з дипломним проектом було визначено актуальність даної теми та досліджено засоби, що були розроблені раніше для вирішення подібних задач. Програмний продукт, що розробляється дозволяє пройти процес навчання моделі самостійно, отримати оцінку кожної із моделей машинного навчання та обрати алгоритм, що найкраще підходить для вирішення конкретної задачі. Таким чином можна зробити висновок що розробка даного програмного продукту є актуальною,

Окрім цього було описано процес діяльності, який автоматизується, визначено, які етапи потрібно пройти, аби розв’язати задачу, наведено коротку характеристику методів, якими пропонується вирішувати поставлену задачу. Було побудовано діаграму діяльності та наведено її детальний опис.

Також, описано функціональну модель системи та визначено функції користувача. Сформовано призначення розробки, встановлено мету та визначено задачі, які необхідно вирішити для досягнення мети.

На наступному етапі були визначені вхідні та вихідні дані системи, що розробляється.

Надано детальний опис вхідних даних. задачі було формалізовано саму задачі та наведено її основні атрибути. Визначено математичний запис задачі. Також, було наведено опис методів розв’язання задачі, наведено коротку характеристику кожного алгоритму. Встановлено переваги та недоліки

алгоритмів. Окрім цього було наведено математичне обґрунтування кожного із методів.

Було розглянуто засоби розробки програмного забезпечення, Обґрунтовано вибір мови програмування Python, а також системи управління базами даних PostgreSQL.

Окрім цього були виявлені загальні вимоги до програмного забезпечення, описано бібліотеки, що використовуватимуться при розробці програмного продукту.

Було описано архітектуру програмного забезпечення. Спроектовано діаграми класів, послідовності та компонентів і наведено їх детальний опис.

Також наведено специфікацію функцій, що використовувалися при створенні програмного продукту та описано звіти, що генеруються в результаті роботи програми.

В процесі тестування була перевірена уся функціональність системи, наведений перелік випробувань основних функціональних можливостей.

Було встановлено, що функціонал розробленого веб-застосунок відповідає встановленим вимогам. Варто зауважити, що всі функції про які було заявлено в розділі 1.1.2 Опис функціональної моделі були розроблені в рамках системи.

Використовуючи розроблену систему класифікації текстів кожен може створити моделі машинного навчання на основі обраних алгоритмів, оцінити ефективності моделей і обрати конкретну модель для вирішення поставленої задачі.

Отже, цілі, що були поставлені на початковому етапі проектування системи класифікації текстів методами обробки природної мови та машинного навчання були досягненими в ході виконання дипломного проекту.

## ПЕРЕЛІК ПОСИЛАНЬ

1. Raymond J. Mining Knowledge from Text Using Information Extraction / J. Raymond, B. Razvan. // Department of Computer Sciences, University of Texas at Austin. – С. 1–10.

2. Vandana K. Text classification and classifiers / Korde Sardar Vandana. // National Institute of Technology, Surat.

3. Словник української мови – Київ, 1978. – (Том 9).

4. Daniel Jurafsky. Classification: Naive Bayes, Logistic Regression, Sentiment / Daniel Jurafsky, James H. Martin // Speech and Language Processing / Daniel Jurafsky, James H. Martin., 2015. – С. 1–28.

5. J Korean Acad Nurs. An Introduction to Logistic Regression: From Basic Concepts to Interpretation with Particular Attention to Nursing Domain / J Korean Acad Nurs. // College of Nursing and System Biomedical Informatics National Core Research Center, Seoul National University, Seoul, Korea.

6. Manabu Sassano. Virtual Examples for Text Classification with Support Vector Machines / Manabu Sassano. // Fujitsu Laboratories Ltd, Kawasaki 211-8588, Japan.

7. Barry de Ville. Decision Trees for Business Intelligence and Data Mining: Using SAS Enterprise Miner / Barry de Ville. – Cary, NC, USA: SAS Institute Inc., 2006.

8. Kevin P. Murphy. Naive Bayes classifier / Kevin P. Murphy. – Vancouver, Canada: Department of Computer Science, University of British Columbia, 2006.

9. Peter M. Lee. Bayesian Statistics: An Introduction, 4th Edition / Peter M. Lee. – 486 с.

10. Mark Lutz. Learning Python / Mark Lutz., 2004. – 552 с.

11. PostgreSQL 10.7 Documentation. // The PostgreSQL Global Development Group. – 2019.

**Додаток А**

***Система класифікації текстів методами обробки природної мови та машинного навчання***

(Найменування програми (документа))

*DVD-R*

(Вид носія даних)

*143 арк, 424 Кб*

(Обсяг програми (документа) , арк.,) Кб)

Київ – 2019 року

Додаток А

```
from flask import Flask, render_template, flash, url_for, request, redirect
from werkzeug import secure_filename
import os
from flask_bootstrap import Bootstrap
import pandas as pd
import numpy as np
import re
import nltk
import time
import csv
#list of irrelevant words (stop-words)
from nltk.corpus import stopwords
#a class for taking the words' roots (loved -> love, systems -> setem e.g.)
from nltk.stem import WordNetLemmatizer
from sklearn.externals import joblib #helps in saving ml models
import matplotlib as plt
import psycopg2
import datetime
import pickle

#-----GLOBAL VARIABLES------
#indexes for the columns with texts and categories
category_column = 0
text_column = 0
#name of the file with dataset
flename = ''
#available categories
categories = ''

app = Flask(__name__)
APP_ROOT = os.path.dirname(os.path.abspath(__file__))

@app.route('/')
def main():
    flash('sdflkdshfjh')
    return render_template('index.html')

@app.route('/newmodel')
def newmodel():
    return render_template('newmodel.html')

def showdataset(filename):
    dataset = pd.read_csv("C:/_Data/Diploma/Datasets/"+filename)
    #dataset = pd.read_csv("C:/_Data/Diploma/Datasets/bbc.csv")
    showdataset.dataset = dataset
    #Get the list of available columns in the dataset
```

```
    columns_list = []
    for i in dataset.columns.values:
        columns_list.append(i)
    n=5 #number of rows to be shown
    first = []
    for i in range(n):
        first.append([dataset[columns_list[0]][i],dataset[columns_list[1]][i]])
    return first

@app.route("/upload_file", methods=['POST'])
def upload_file():
    global filename
    file = request.files['file']
    #flash(file.filename)
    filename = file.filename
    first = showdataset(filename)
    return render_template('newmodel.html', first=first, sh = bool(first))

@app.route("/define_lables", methods=['POST'])
def define_lables():
    global text_column
    global category_column
    global filename

    first = showdataset(filename)

    if request.method == 'POST':
        label_choice = request.form.get('labels')
        text_choice = request.form.get('texts')
        if (label_choice == text_choice):
            label_error = "Please make sure that the column with texts is differ from the column with
labels"
            return render_template('newmodel.html', first=first, shtx = bool(text_choice), sh = True,
label_error=label_error)
        print(label_choice)
        print(text_choice)

        if int(text_choice) == 2:
          category_column = 0
          text_column = 1
        else:
            category_column = 1
            text_column = 0

        flash("category:"+str(label_choice))
        flash("text:"+str(text_choice))
        return redirect(url_for('choose_algorithms'))
    return render_template('newmodel.html', first=first, sh = True)
```

```
@app.route("/choose_algorithms", methods=['GET','POST'])
def choose_algorithms():
    choose_algorithms.algorithms = []
    if request.method == 'POST':
        choose_algorithms.algorithms = request.form.getlist('algorithm')
        return redirect(url_for('creating_models'))
    return render_template('choose_algorithms.html')

corpus = []
maxfeatures = 0
def text_preprocessing(dataset):
    global corpus
    global text_column
    for i in range(len(dataset.index)):
        #remove non-letter symbols
        article = re.sub('[^a-zA-Z]', ' ', dataset[list(dataset.columns.values)[text_column]][i])
        #replace upper cases with lower cases
        article = article.lower()
        #split string to a list
        article = article.split()
        #remove non-significant words (the, that, and, in ...) and get roots of the words
        wnl = WordNetLemmatizer()
        article = [wnl.lemmatize(word) for word in article if not word in
set(stopwords.words('english'))]
        #create a string
        article = ' '.join(article)
        #add article to a corpus
        corpus.append(article)
        print(i)
    return corpus

classifier_LR = False
classifier_NB = False
classifier_SVC = False
classifier_DT = False

def write_model_info(dataset_name, algorithm_name, accuracy, trains, tests, model_filename):
    global categories

    #WRITE TO THE FILE
    model_info_filename = "C:/_Data/Diploma/Models
Info/"+dataset_name.replace('.csv','')+"_"+algorithm_name+".txt"
    f = open(model_info_filename, "w")
    f.write("Here you can find important information about the created model!\n")
    f.write("Model name: "+dataset_name.replace('.csv','')+"_"+algorithm_name+";\n")
    if algorithm_name == 'lr':
        algorithm = "Logistic Regression"
    if algorithm_name == 'nb':
```

```
        algorithm = "Naive Bayes"
    if algorithm_name == 'sv':
        algorithm = "Support Vector Machine"
    if algorithm_name == 'dt':
        algorithm = "Decision Tree"
    f.write("Algorithm: " + algorithm+";\n")
    date_object = datetime.date.today()
    f.write("Date of creation: " + str(date_object) + ";\n")
    f.write("Accuracy: " + str(accuracy)+";\n")
    f.write("Dataset: "+ dataset_name+";\n")
    f.write("Number of the records in the dataset: " + str(trains+tests)+";\n")
    f.write("Number of the train records in the dataset: " + str(trains)+";\n")
    f.write("Number of the test records in the dataset: " + str(tests) +";\n")
    f.write("Number of the categories in the dataset: " + str(len(categories))+";\n")
    f.write("Categories: " + str(categories) +";\n")
    f.close()

    #WRITE TO THE DATABASE
    conn = psycopg2.connect(
        database = "Models",
        user = "Dmytro",
        password = "1234",
        host = "localhost",
        port = "5432")
    cur = conn.cursor()

    cat = str(categories).replace(',',';').replace("'","~")
    #Write to the datasets table
    query_dataset = "insert into dataset_info select (nextval('datasets_sequence')),
'C:/_Data/Diploma/Datasets/"+dataset_name+"',"+str(trains+tests)+", "+str(trains)+ ", "+
str(tests)+ ", "+ str(len(categories))+", '"+str(cat)+"' where not exists (select file_with_dataset
FROM dataset_info WHERE file_with_dataset =
'C:/_Data/Diploma/Datasets/"+dataset_name+"');"
    cur.execute(query_dataset)

    #Get the index of the last added dataset
    cur.execute("select id from dataset_info order by id desc limit 1")
    ind = cur.fetchall()[0][0]

    #Write to the models table
    query_model = "insert into model_info values
(nextval('models_sequence'),"+str(ind)+",'"+dataset_name.replace('.csv','')+"_"+algorithm_name
+"','"+algorithm+"', '"+str(datetime.date.today())+"', '"+dataset_name+"',
"+str(accuracy)+",'"+model_filename+"','C:/_Data/Diploma/Models
Info/"+dataset_name.replace('.csv','')+"_"+algorithm_name+".txt')"
    cur.execute(query_model)
    conn.commit()
    conn.close()
```

```
def save_info_to_db(dataset_name,algorithm_name,accuracy,model_filename):
    conn = psycopg2.connect(
        database = "Models",
        user = "Dmytro",
        password = "1234",
        host = "localhost",
        port = "5432")
    cur = conn.cursor()
    cur.execute("INSERT INTO MODEL_INFO VALUES
(nextval('models_sequence'),'"+dataset_name.replace('.csv','')+"_"+algorithm_name +
"','"+str(accuracy)+"','"+str(datetime.date.today())+"','C:/_Data/Diploma/Models
Info/"+dataset_name.replace('.csv','')+"_"+algorithm_name+".txt','"+"'C:/_Data/Diploma/Dataset
s/"+dataset_name+"','"+model_filename+"')")
    conn.commit()
    conn.close()

@app.route("/creating_models", methods=['GET', 'POST'])
def creating_models():
    dataset = showdataset.dataset
    global corpus
    global filename
    global categories
    corpus = []
    #check if the corpus was made before
    corpus_filename = "C:/_Data/Diploma/Datasets/corpus/"+filename.replace('.csv',
'_corpus.csvc')
    #dataset = pd.read_csv("C:/_Data/Diploma/Datasets/bbc.csv")

    create_corpus = False

    #indicatess whether models were created
    lr_created = False
    nb_created= False
    sv_created = False
    dt_created = False

    try: #if the corpus exists
      corpus = pd.read_csv(corpus_filename)
      #corpus1 = pd.read_csv("C:\_Data\Diploma\Datasets\corpus\expressions_corpus.csvc")
      print("text_column:")
      print(text_column)
      corpus = corpus.loc[:,list(dataset)[text_column]].values.tolist()
      #corpus1 = corpus.loc[:,list(dataset)[0]].values.tolist()
      print("Corpus is ready for training models.")
    except IOError:
      create_corpus = True
```

```
        print ("Corpus for this dataset does not appear to exist. Creating corpus...")
        timer = time.time()+0.4*len(dataset.index)
        flash("ETA for ending is: "+ time.strftime("%I %M %p",time.localtime(timer)))
        if 'textprocessing' in request.form:
            corpus = text_preprocessing(dataset)
            with   open("C:/_Data/Diploma/Datasets/corpus/"+filename.replace('.csv',   '_corpus.csvc'),
'w') as f:
                writer = csv.writer(f, lineterminator='\n')
                writer.writerow([list(dataset)[text_column]])
                for val in corpus:
                    writer.writerow([val])

    if(len(corpus) != 0): #if corpus is not empty
        #Creating the Bag of Words model
        from sklearn.feature_extraction.text import CountVectorizer
        global cv
        cv = CountVectorizer(max_features = 20000)
        print("filename")
        print(filename)
        X = cv.fit_transform(corpus)
        #X = cv.fit_transform(corpus)

        #Save Countvectorizer
        pickle.dump(cv,          open("C:/_Data/Diploma/Countvectorizers/"+filename.replace('.csv',
'.pickle'), "wb"))

        global maxfeatures
        #maxfeatures = len(X[0])
        maxfeatures = 0
        y = dataset.iloc[:, category_column].values
        categories = set(y)

        print("X:\n")
        print(X)
        print("\n\n\n\n\n\n\n\n")
        print("y:\n")
        print(y)
        print("\n\n\n\n\n\n\n\n")
        print(filename)
        from sklearn.model_selection import train_test_split
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 42)


        from sklearn.metrics import confusion_matrix, accuracy_score

        if 'lr' in choose_algorithms.algorithms:
            model_filename = 'C:/_Data/Diploma/Models/'+filename.replace('.csv', '_model.lr')
            global classifier_LR
```

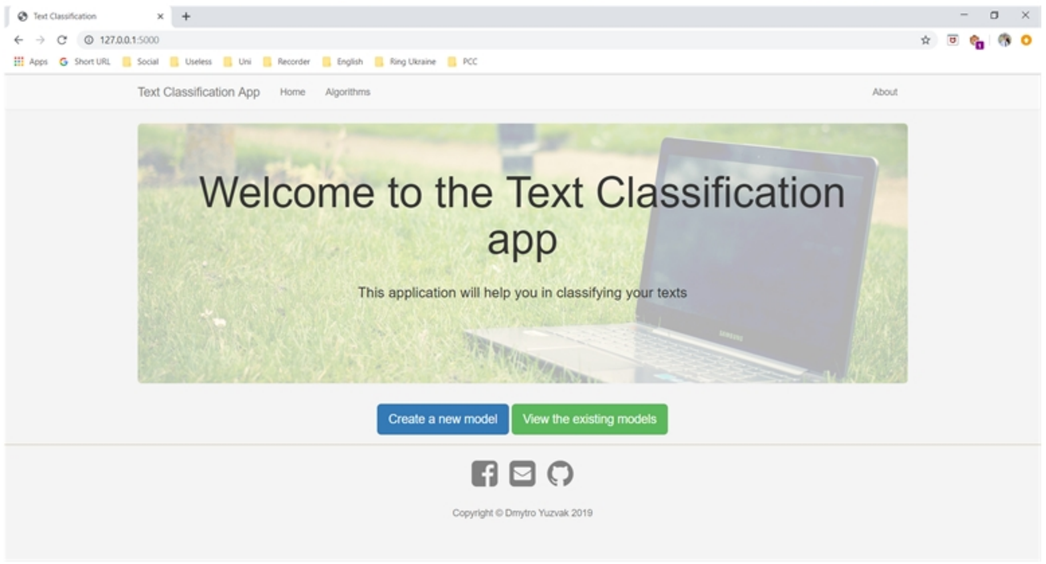
```
        lr_created = True
        try:
            classifier_LR = joblib.load(model_filename)
            print("LR model loaded")
        except IOError:
            from sklearn.linear_model import LogisticRegression
            classifier_LR = LogisticRegression(random_state = 42)
            classifier_LR.fit(X_train, y_train)
            print("LR model created")
            joblib.dump(classifier_LR, model_filename)
            y_pred = classifier_LR.predict(X_test)
            #write_model_info(filename, "lr", accuracy_score(y_test, y_pred), len(X_train), len(X_test), model_filename)
            write_model_info(filename, "lr", accuracy_score(y_test, y_pred), len(X_train.toarray()), len(X_test.toarray()), model_filename)
            #save_info_to_db(filename, "lr", accuracy_score(y_test, y_pred),model_filename)
        y_pred = classifier_LR.predict(X_test)
        flash("LR: "+str(accuracy_score(y_test, y_pred)))
        print("The accuracy of Logistic Regression Algorithm is " + str(accuracy_score(y_test, y_pred)))
        #cm = confusion_matrix(y_test, y_pred)

    X_train = X_train.toarray()
    X_test = X_test.toarray()

    if 'nb' in choose_algorithms.algorithms:
        model_filename = 'C:/_Data/Diploma/Models/'+filename.replace('.csv', '_model.nb')
        global classifier_NB
        nb_created = True
        try:
            classifier_NB = joblib.load(model_filename)
            print("NB model loaded")
        except IOError:
            from sklearn.naive_bayes import GaussianNB
            classifier_NB = GaussianNB()




            classifier_NB.fit(X_train, y_train)
            print("NB model created")
            joblib.dump(classifier_NB, model_filename)
            y_pred = classifier_NB.predict(X_test)

            write_model_info(filename, "nb", accuracy_score(y_test, y_pred), len(X_train), len(X_test), model_filename)
            #save_info_to_db(filename, "nb", accuracy_score(y_test, y_pred),model_filename)
```

```
        y_pred = classifier_NB.predict(X_test)
        flash("NB: "+str(accuracy_score(y_test, y_pred)))
        print("The accuracy of Naive Bayes Algorithm is " + str(accuracy_score(y_test, y_pred)))
        #cm = confusion_matrix(y_test, y_pred)

    if 'sv' in choose_algorithms.algorithms:
        global classifier_SVC
        model_filename = 'C:/_Data/Diploma/Models/'+filename.replace('.csv', '_model.sv')
        sv_created = True
        try:
            classifier_SVC = joblib.load(model_filename)
            print("SVM model loaded")
        except IOError:
            from sklearn.svm import LinearSVC
            classifier_SVC = LinearSVC()



            classifier_SVC.fit(X_train, y_train)
            print("SVM model created")
            joblib.dump(classifier_LR, model_filename)
            y_pred = classifier_SVC.predict(X_test)
            write_model_info(filename, "sv", accuracy_score(y_test, y_pred), len(X_train),
len(X_test), model_filename)
            #save_info_to_db(filename, "sv", accuracy_score(y_test, y_pred),model_filename)
        y_pred = classifier_SVC.predict(X_test)
        flash("SVM: "+str(accuracy_score(y_test, y_pred)))
        print("The accuracy of Support Vector Machine Algorithm is " +
str(accuracy_score(y_test, y_pred)))
        #cm = confusion_matrix(y_test, y_pred)

    if 'dt' in choose_algorithms.algorithms:
        global classifier_DT
        model_filename = 'C:/_Data/Diploma/Models/'+filename.replace('.csv', '_model.dt')
        dt_created = True
        try:
          classifier_DT = joblib.load(model_filename)
          print("DT model loaded")
        except IOError:
            from sklearn.tree import DecisionTreeClassifier
            classifier_DT = DecisionTreeClassifier(criterion = 'entropy', random_state=42)



            classifier_DT.fit(X_train, y_train)
            print("DT model created")
            joblib.dump(classifier_DT, model_filename)
```

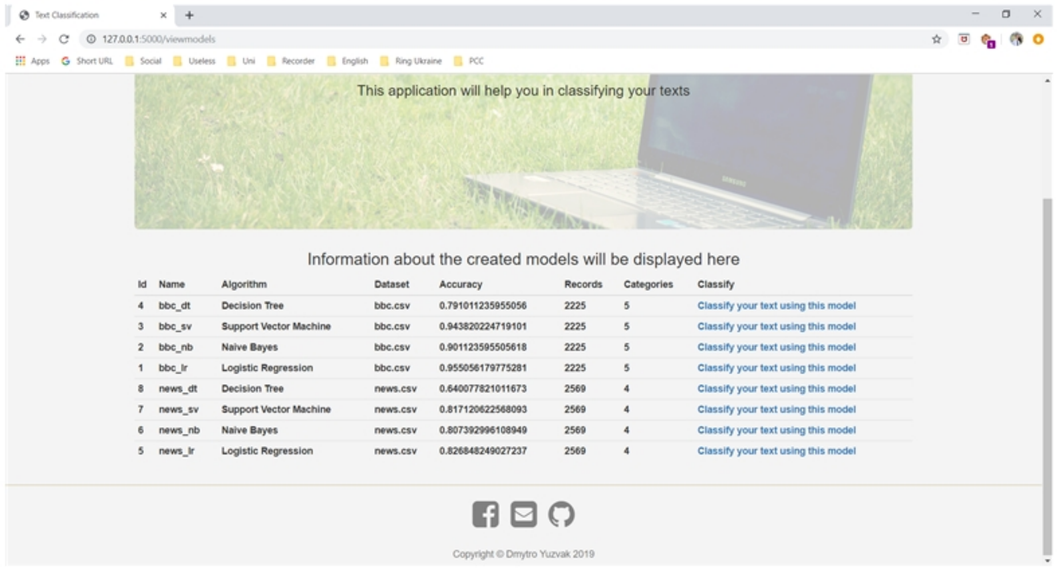
```
            y_pred = classifier_DT.predict(X_test)
            write_model_info(filename, "dt", accuracy_score(y_test, y_pred), len(X_train), len(X_test), model_filename)
            #save_info_to_db(filename, "dt", accuracy_score(y_test, y_pred),model_filename)
        y_pred = classifier_DT.predict(X_test)
        flash("DT: "+str(accuracy_score(y_test, y_pred)))
        print("The accuracy of Desicion Tree Algorithm is " + str(accuracy_score(y_test, y_pred)))

        #cm = confusion_matrix(y_test, y_pred)


    return redirect(url_for('model_info'))

  return render_template("creating_models.html", create_corpus=create_corpus, lr_created=lr_created,nb_created=nb_created,sv_created=sv_created,dt_created=dt_created)

model = False

@app.route('/model_info', methods=['GET','POST'])
def model_info():

  global filename

  #indicatess whether models were created
  lr_created = False
  nb_created= False
  sv_created = False
  dt_created = False

  content_lr = ""
  content_nb = ""
  content_sv = ""
  content_dt = ""

  file_with_info = "C:/_Data/Diploma/Models Info/"


  if 'lr' in choose_algorithms.algorithms:
      lr_created = True
      info = open(file_with_info+filename.replace(".csv", "_lr.txt") , 'r+')
      content_lr = info.read()
      info.close()
  if 'nb' in choose_algorithms.algorithms:
      nb_created = True
      info = open(file_with_info+filename.replace(".csv", "_nb.txt") , 'r+')
      content_nb = info.read()
      info.close()
```

```
    if 'sv' in choose_algorithms.algorithms:
            sv_created = True
            info = open(file_with_info+filename.replace(".csv", "_sv.txt") , 'r+')
            content_sv = info.read()
            info.close()
    if 'dt' in choose_algorithms.algorithms:
            dt_created = True
            info = open(file_with_info+filename.replace(".csv", "_dt.txt") , 'r+')
            content_dt = info.read()
            info.close()

    if request.method == 'POST':
        model_type = request.form['model_type']



        global model

        if (str(model_type) == 'Classify your text with the Logistic Regression Model'):
            model = 'lr'

        if (str(model_type) == 'Classify your text with the Naive Bayes Model'):
            model = 'nb'

        if (str(model_type) == 'Classify your text with the Support Vector Machine Model'):
            model = 'svm'

        if (str(model_type) == 'Classify your text with the Decision Tree Model'):
            model = 'dt'

        if (str(model_type) == 'Classify your text with all the Models'):
            model = 'all'
        return redirect(url_for('model_classify'))
    return                                                  render_template('model_info.html',
lr_created=lr_created,nb_created=nb_created,sv_created=sv_created,dt_created=dt_created,
content_lr = content_lr, content_nb = content_nb, content_sv = content_sv, content_dt =
content_d
```

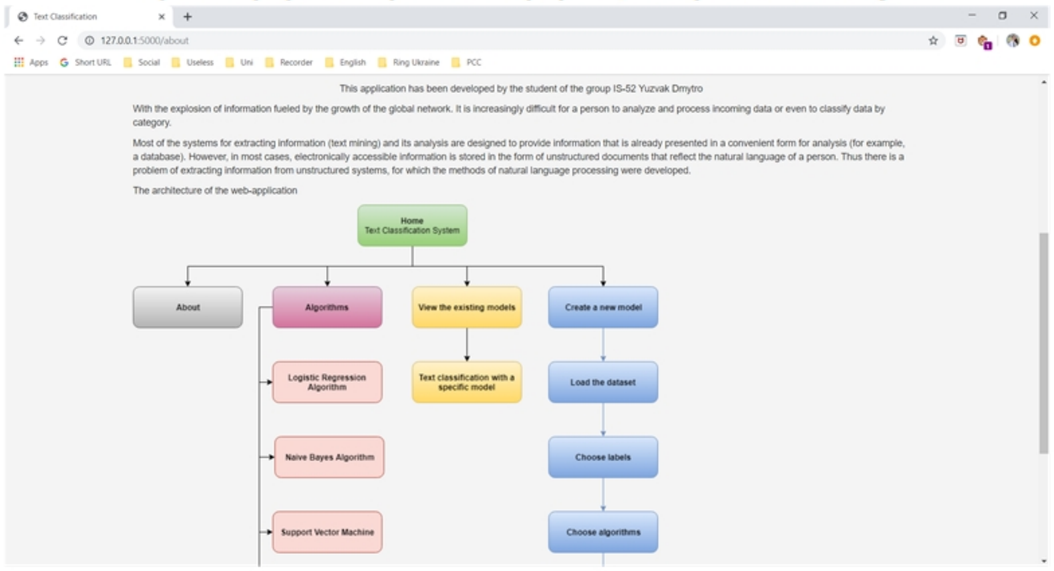
## Екранна форма головної сторінки веб-застосунку



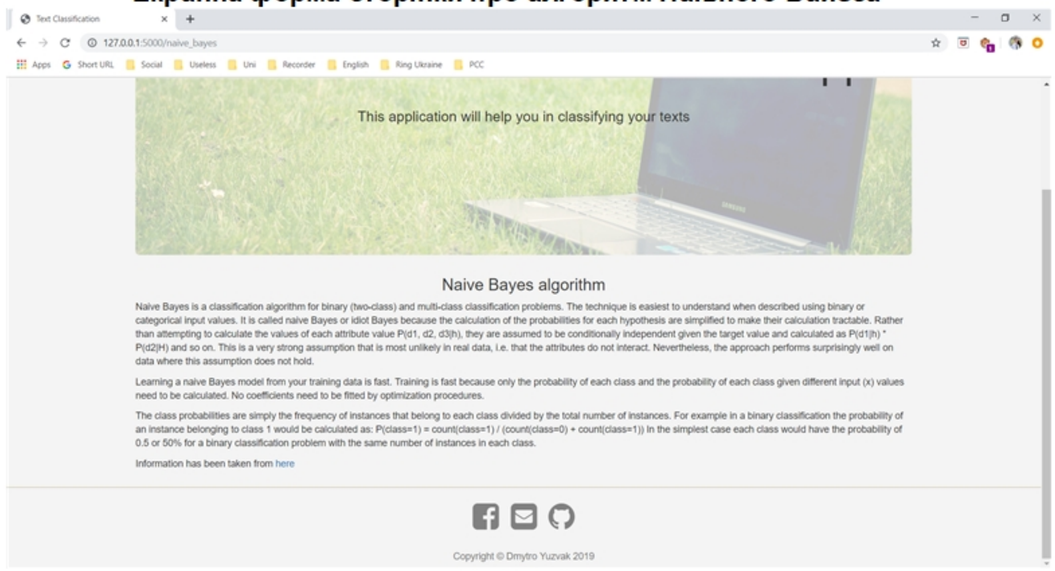

## Екранна форма сторінки з інформацією про створені моделі машинного навчання




## Екранна форма сторінки із інформацією про веб-застосунок



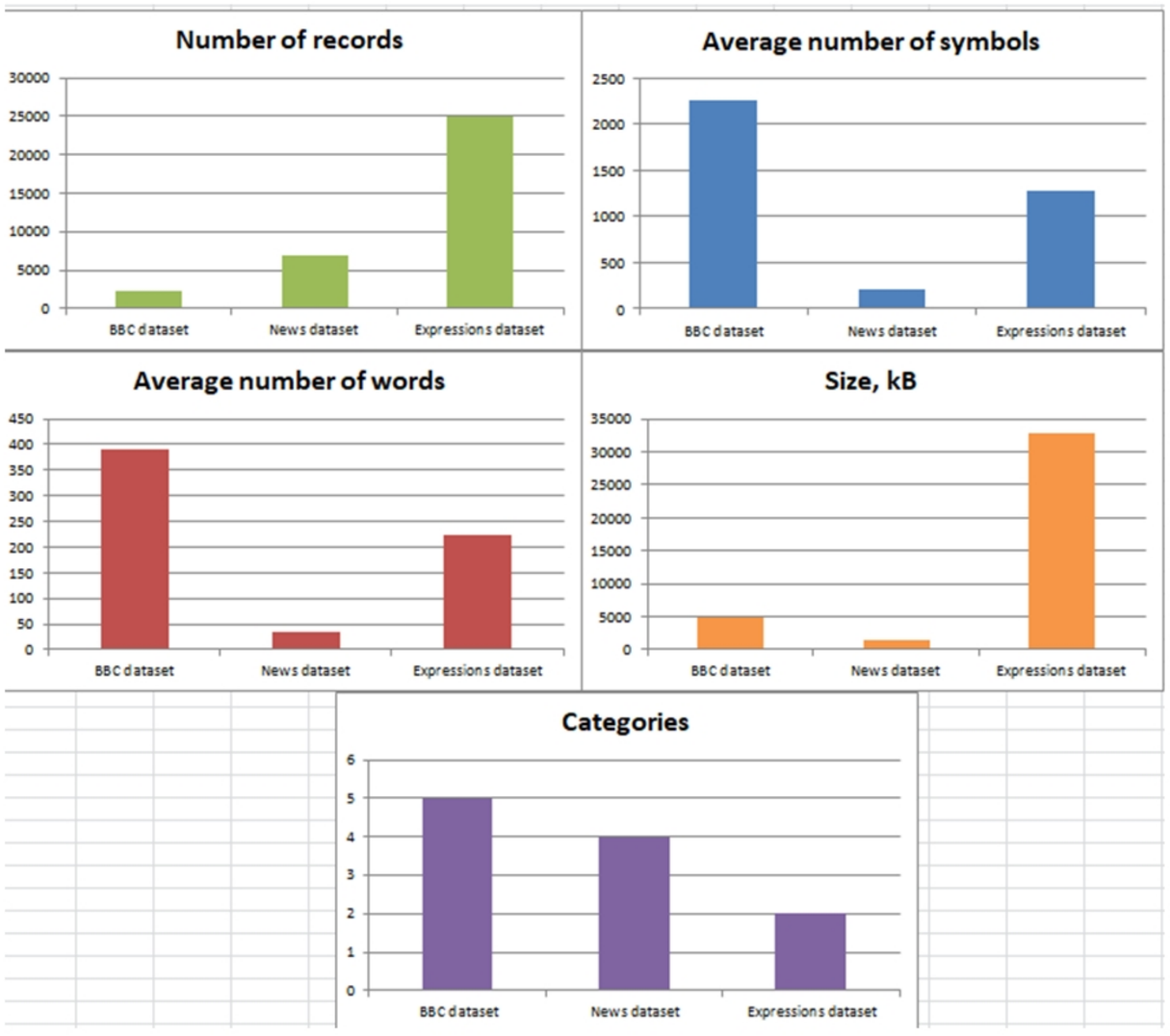

## Екранна форма сторінки про алгоритм Наївного Байєса



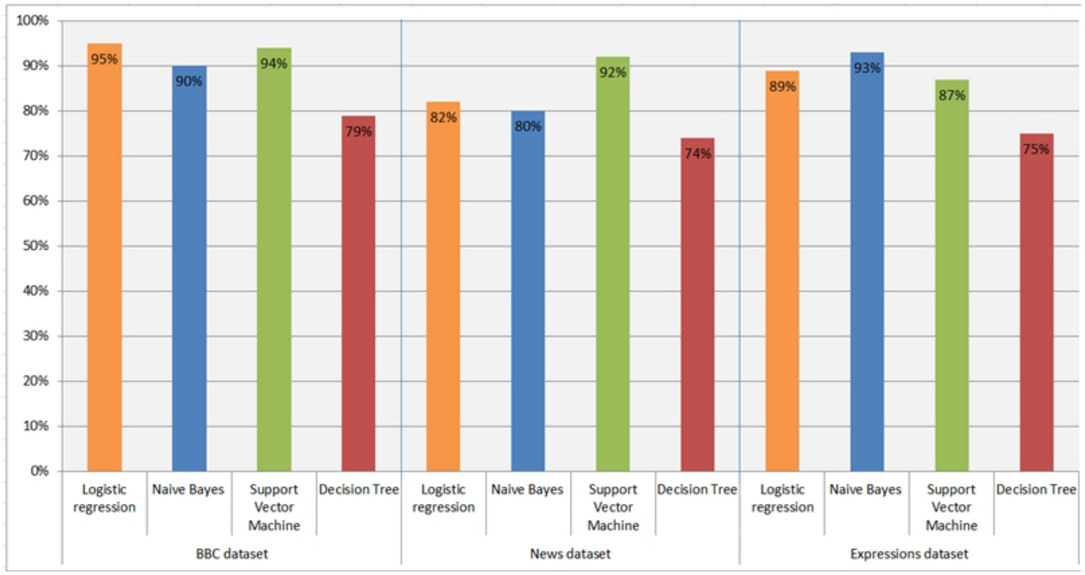

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | ДП ІС-5226.1181-с.КЕ | |
| Зм. | Арк. | № документа | Підпис | Дата | Креслення вигляду екранних форм | Літера / Маса / Масштаб |
| Розробив | | Юзьвак Д.Ю. | | | | |
| Перевірив | | Сперкач М.О. | | | | Аркуш 1 / Аркушів 2 |
| Т. кон. | | | | | | |
| Н. кон. | | Халус О.А. | | | Система класифікації текстів методами обробки природної мови та машинного навчання | КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52 |
| Затвердив | | Сперкач М.О. | | | | |

# Рішення з математичного забезпечення

**Порівняльна характеристика дата сетів**




**Порівняльна характеристика ефективностей моделей машинного навчання**



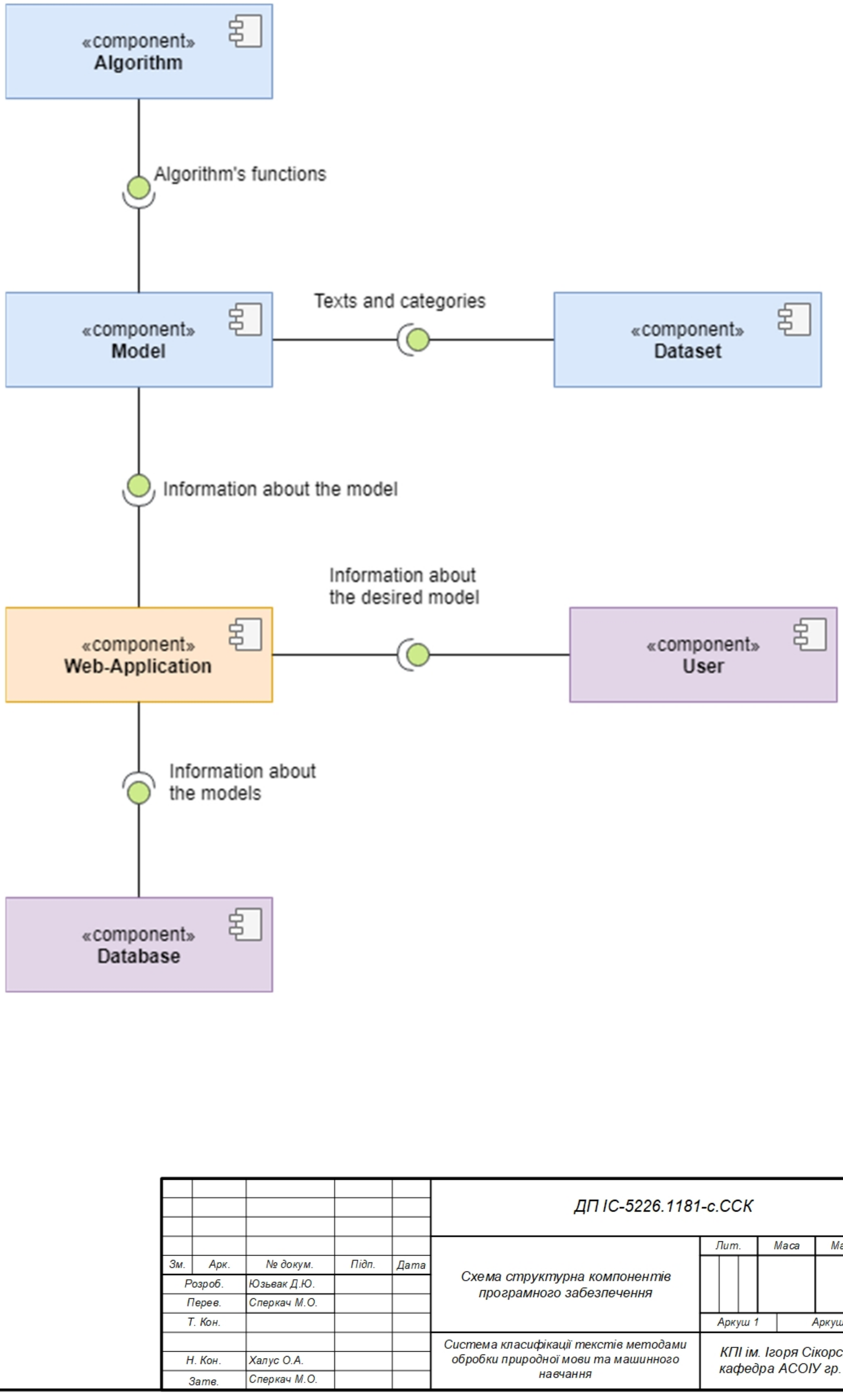

Демонстраційний плакат до дипломного проекту

„Система класифікації текстів методами обробки природної мови та машинного навчання ”

Виконав студент гр. ІС-52 Юзьвак Д.Ю.

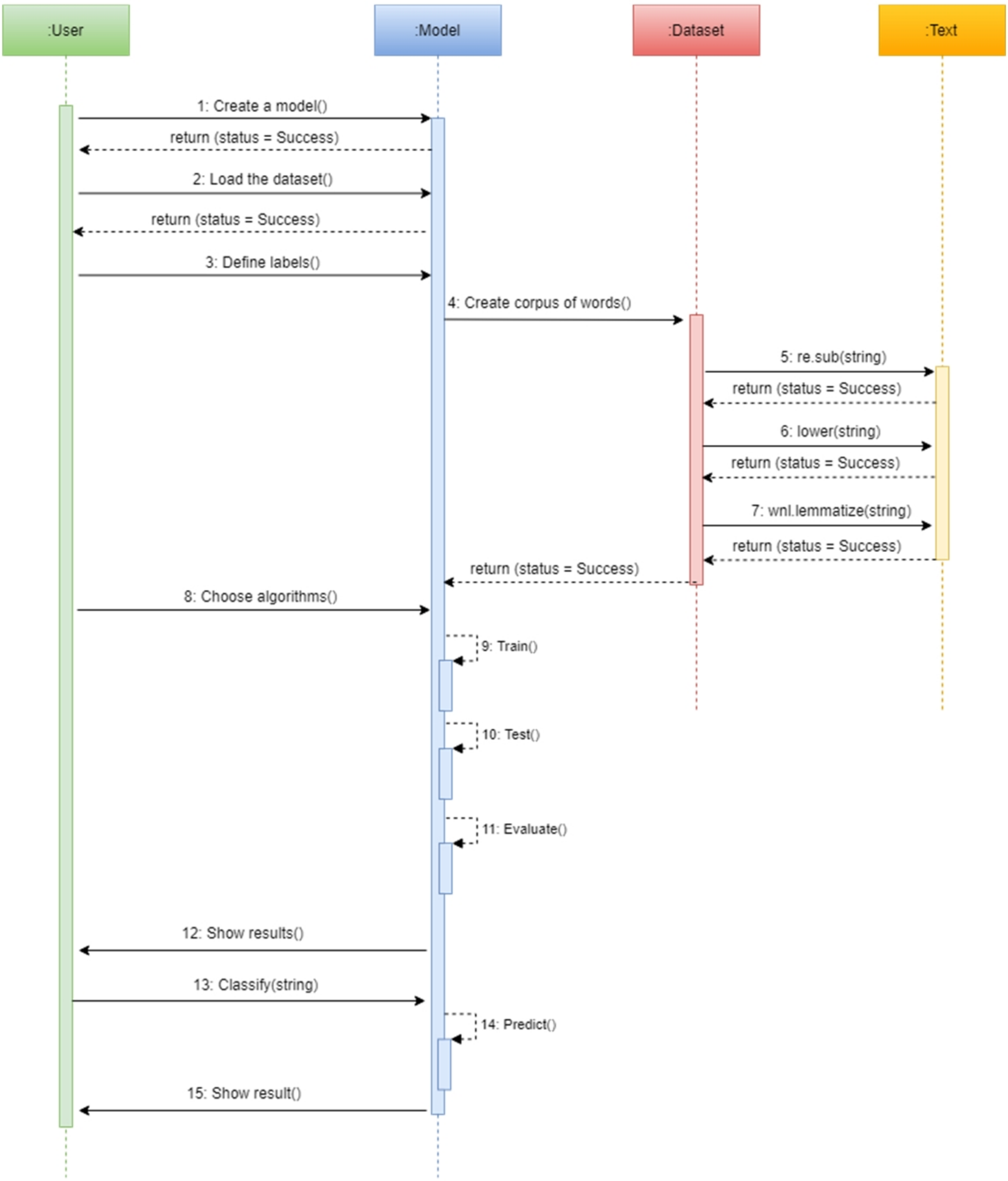
Керівник ДП Сперкач М.О.

ДП ІС-5226.1181-с.ССК



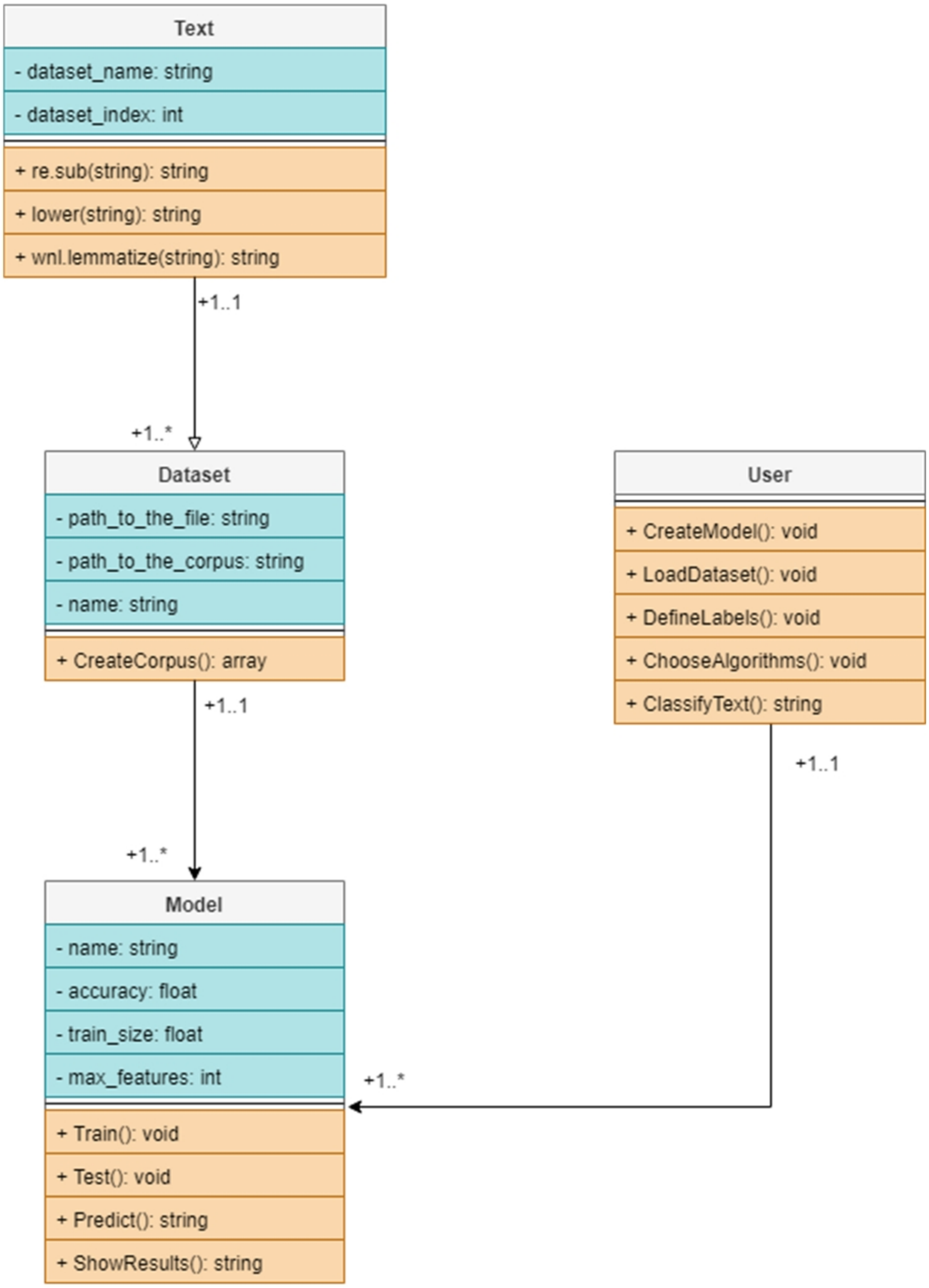

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | ДП ІС-5226.1181-с.ССК | |
| | | | | | Схема структурна компонентів програмного забезпечення | Лит. / Маса / Масштаб |
| Зм. | Арк. | № докум. | Підп. | Дата | | Аркуш 1 / Аркушів 1 |
| Розроб. | | Юзьвак Д.Ю. | | | Система класифікації текстів методами обробки природної мови та машинного навчання | КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52 |
| Перев. | | Сперкач М.О. | | | | |
| Т. Кон. | | | | | | |
| Н. Кон. | | Халус О.А. | | | | |
| Затв. | | Сперкач М.О. | | | | |

ДП ІС-5226.1181-с.ССП



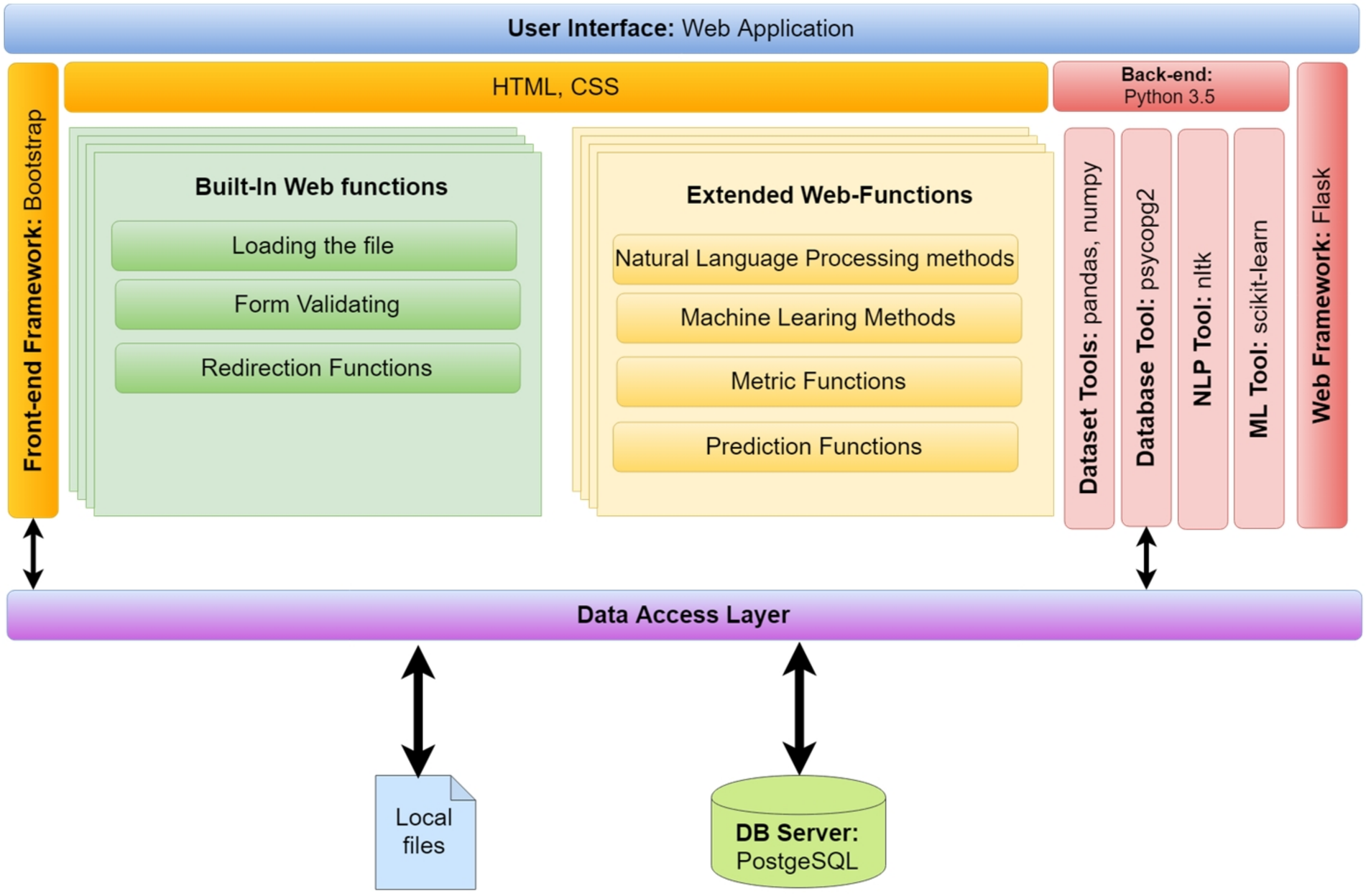

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | ДП ІС-5226.1181-с.ССП | |
| Зм. | Арк. | № докум. | Підп. | Дата | Схема структурна послідовності програмного забезпечення | Лит. / Маса / Масштаб |
| Розроб. | | Юзьвак Д.Ю. | | | | |
| Перев. | | Сперкач М.О. | | | | Аркуш 1 / Аркушів 1 |
| Т. Кон. | | | | | Система класифікації текстів методами обробки природної мови та машинного навчання | КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52 |
| Н. Кон. | | Халус О.А. | | | | |
| Затв. | | Сперкач М.О. | | | | |

ДП ІС-5226.1181-с.ССК

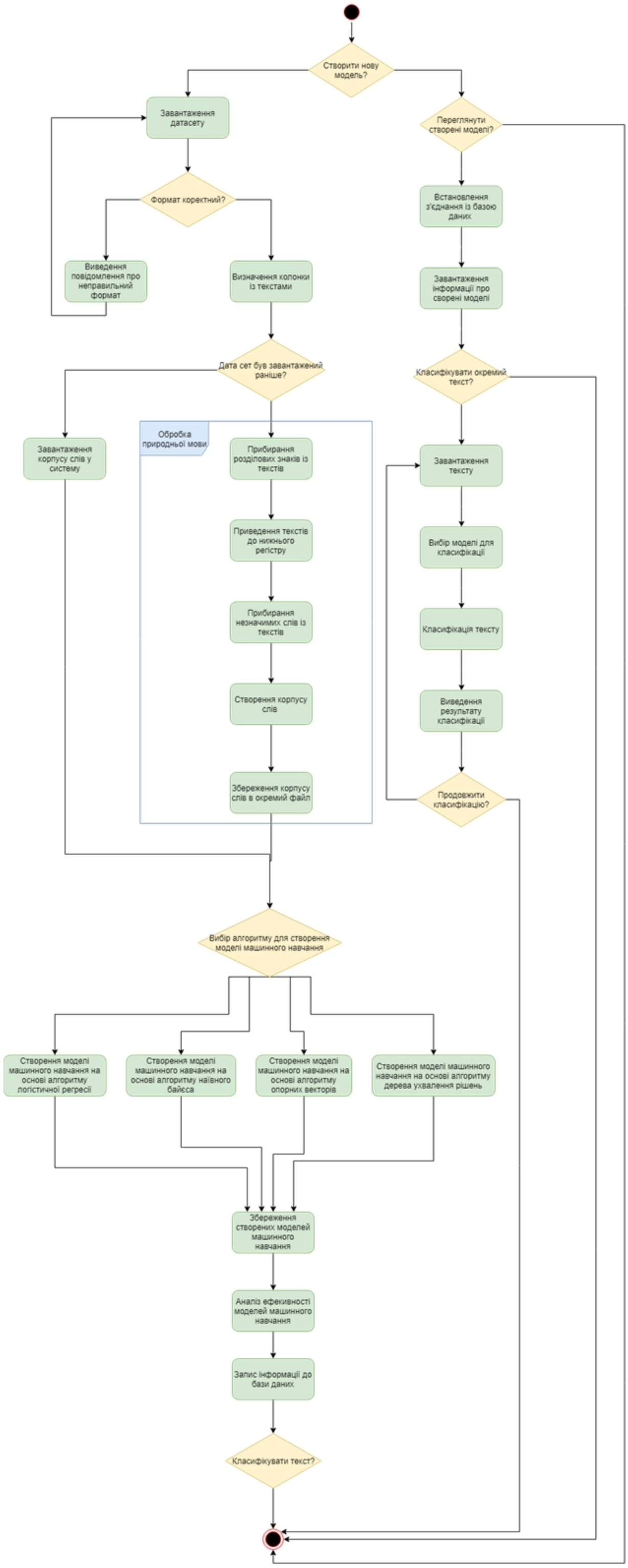



| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | ДП ІС-5226.1181-с.ССК | |
| | | | | | Схема структурна класів програмного забезпечення | Лит. / Маса / Масштаб |
| Зм. | Арк. | № докум. | Підп. | Дата | | Аркуш 1 / Аркушів 1 |
| Розроб. | | Юзьвак Д.Ю. | | | Система класифікації текстів методами обробки природної мови та машинного навчання | КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52 |
| Перев. | | Сперкач М.О. | | | | |
| Т. Кон. | | | | | | |
| Н. Кон. | | Халус О.А. | | | | |
| Затв. | | Сперкач М.О. | | | | |

ДП ІС-5226.1181-с.ССА



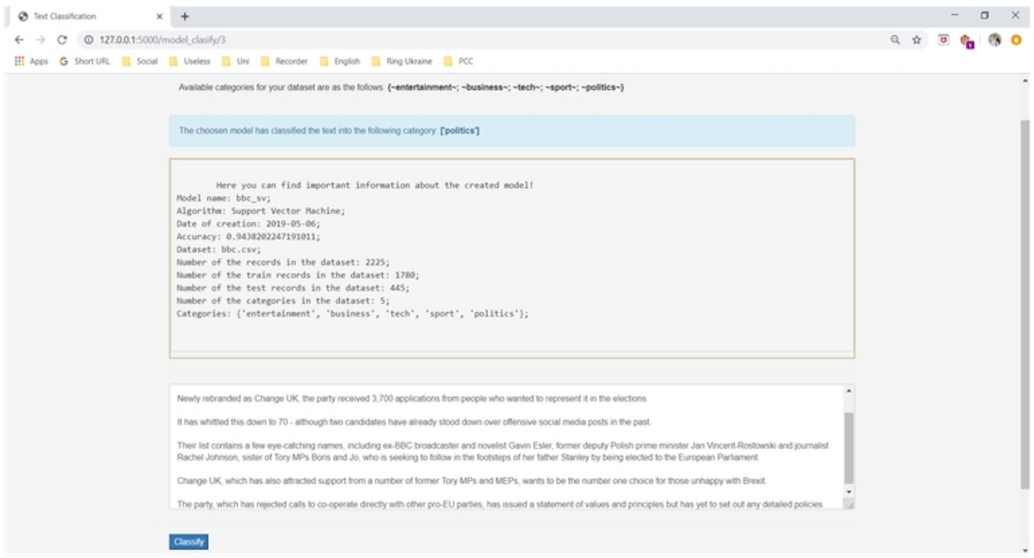

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | ДП ІС-5226.1181-с.ССА | | |
| | | | | | Схема структурна архітектури порграмного забезпечення | Літера | Маса / Масштаб |
| Зм. | Арк. | № документа | Підпис | Дата | | | |
| Розробив | | Юзьвак Д.Ю. | | | | Аркуш 1 | Аркушів 1 |
| Перевірив | | Сперкач М.О. | | | | | |
| Т. кон. | | | | | | | |
| Н. кон. | | Халус О.А. | | | Система класифікації текстів методами обробки природної мови та машинного навчання | КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52 | |
| Затвердив | | Сперкач М.О. | | | | | |



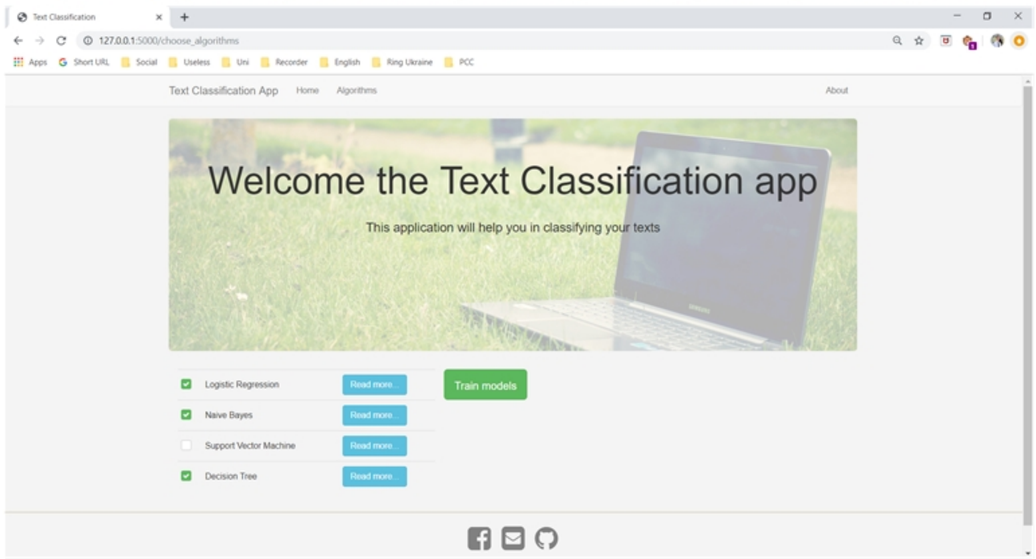

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | ДП ІС-5226.1181-с.ССД | |
| Зм. | Арк. | № докум. | Підп. | Дата | Схема структурна процесу діяльності | Лит. / Маса / Масштаб |
| Розроб. | | Юзьвак Д.Ю. | | | | |
| Перев. | | Сперкач М.О. | | | | Аркуш 1 / Аркушів 1 |
| Т. Кон. | | | | | Система класифікації текстів методами обробки природної мови та машинного навчання | КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52 |
| Н. Кон. | | Халус О.А. | | | | |
| Затв. | | Сперкач М.О. | | | | |

ДП ІС-5226.1181-с.КЕ

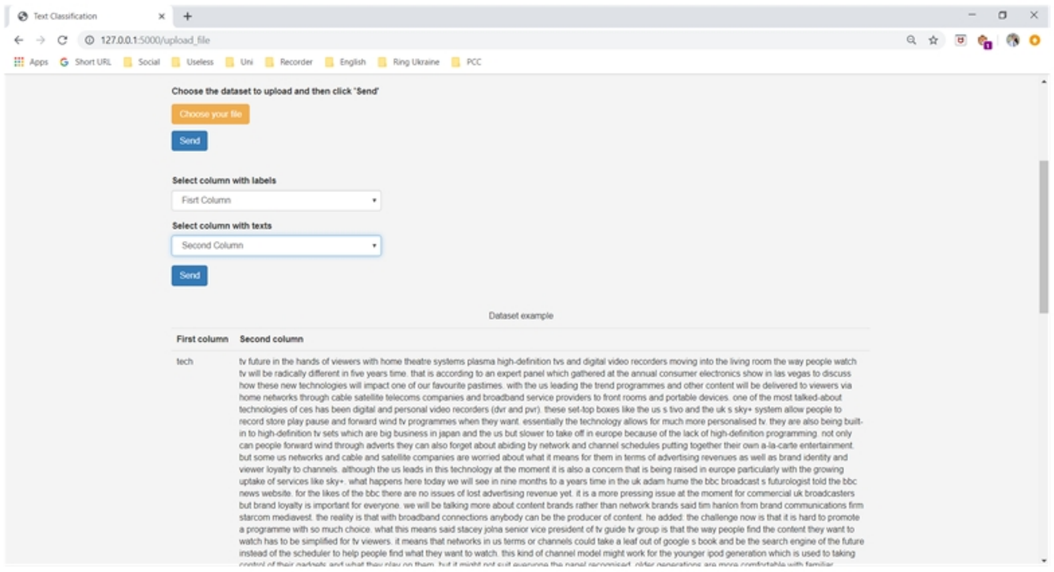
## Екранна форма сторінки із функцією класифікації тексту



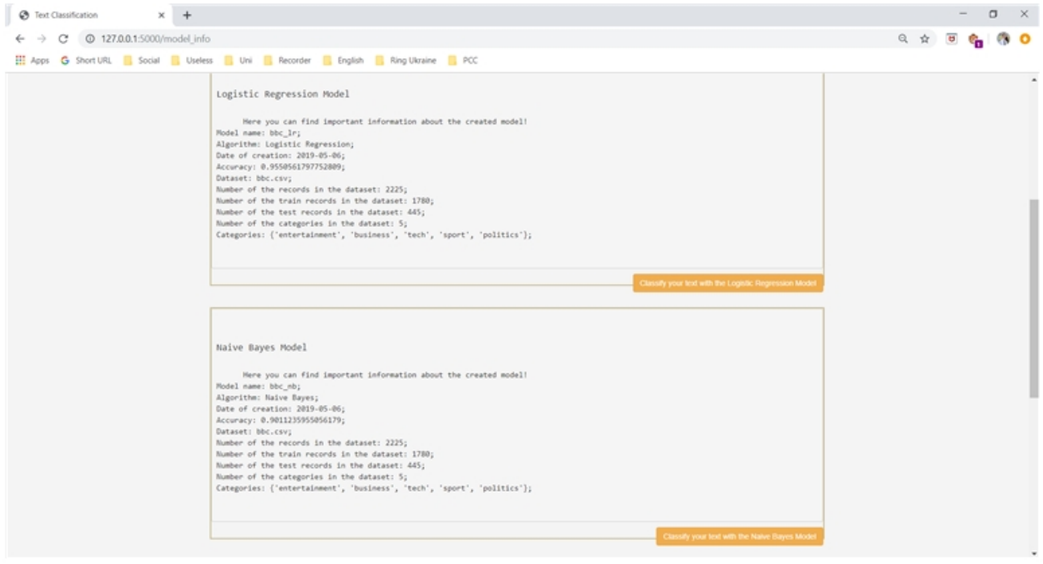

## Екранна форма сторінки із вибором алгоритмів для створення моделей




## Екранна форма сторінки створення нової моделі машинного навчання




## Екранна форма сторінки х відображенням іноформацій про створені моделі




| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | ДП ІС-5226.1181-с.КЕ | |
| | | | | | | |
| | | | | | | |
| Зм. | Арк. | № документа | Підпис | Дата | Креслення вигляду екранних форм | Літера / Маса / Масштаб |
| Розробив | | Юзьвак Д.Ю. | | | | |
| Перевірив | | Сперкач М.О. | | | | Аркуш 2 / Аркушів 2 |
| Т. кон. | | | | | | |
| | | | | | Система класифікації текстів методами обробки природної мови та машинного навчання | КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52 |
| Н. кон. | | Халус О.А. | | | | |
| Затвердив | | Сперкач М.О. | | | | |